

NNN	NNN	EEEEEEEEEEEEEEEE	TTTTTTTTTTTTTTTT	AAAAAAAAAA	CCCCCCCCCCCC	PPPPPPPPPPPP
NNN	NNN	EEEEEEEEEEEEEEEE	TTTTTTTTTTTTTTTT	AAAAAAAAAA	CCCCCCCCCCCC	PPPPPPPPPPPP
NNN	NNN	EEEEEEEEEEEEEEEE	TTTTTTTTTTTTTTTT	AAAAAAAAAA	CCCCCCCCCCCC	PPPPPPPPPPPP
NNN	NNN	EEE	TTT	AAA	CCC	PPP
NNN	NNN	EEE	TTT	AAA	CCC	PPP
NNN	NNN	EEE	TTT	AAA	CCC	PPP
NNNNNN	NNN	EEE	TTT	AAA	CCC	PPP
NNNNNN	NNN	EEE	TTT	AAA	CCC	PPP
NNNNNN	NNN	EEE	TTT	AAA	CCC	PPP
NNN	NNN	EEEEEEEEEEEE	TTT	AAA	CCC	PPP
NNN	NNN	EEEEEEEEEEEE	TTT	AAA	CCC	PPP
NNN	NNN	EEEEEEEEEEEE	TTT	AAA	CCC	PPP
NNN	NNN	EEE	TTT	AAA	CCC	PPP
NNN	NNNNNN	EEE	TTT	AAAAAAAAAAAAAAAA	CCC	PPP
NNN	NNNNNN	EEE	TTT	AAAAAAAAAAAAAAAA	CCC	PPP
NNN	NNNNNN	EEE	TTT	AAAAAAAAAAAAAAAA	CCC	PPP
NNN	NNN	EEE	TTT	AAA	CCC	PPP
NNN	NNN	EEE	TTT	AAA	CCC	PPP
NNN	NNN	EEE	TTT	AAA	CCC	PPP
NNN	NNN	EEE	TTT	AAA	CCC	PPP
NNN	NNN	EEEEEEEEEEEE	TTT	AAA	CCCCCCCCCCCC	PPP
NNN	NNN	EEEEEEEEEEEE	TTT	AAA	CCCCCCCCCCCC	PPP
NNN	NNN	EEEEEEEEEEEE	TTT	AAA	CCCCCCCCCCCC	PPP

-S  
 Ps  
 --  
 NE  
  
 NE  
  
 NE  
  
 NE  
  
 \$R

NN	NN	EEEEEEEEEE	TTTTTTTTTT	PPPPPPPP	RRRRRRRR	000000	CCCCCCCC	RRRRRRRR	EEEEEEEEEE	
NN	NN	EEEEEEEEEE	TTTTTTTTTT	PPPPPPPP	RRRRRRRR	000000	CCCCCCCC	RRRRRRRR	EEEEEEEEEE	
NN	NN	EE	TT	PP	RR	00	CC	RR	EE	
NN	NN	EE	TT	PP	RR	00	CC	RR	EE	
NNNN	NN	EE	TT	PP	RR	00	CC	RR	EE	
NNNN	NN	EE	TT	PP	RR	00	CC	RR	EE	
NN	NN	EEEEEEEE	TT	PPPPPPPP	RRRRRRRR	00	CC	RRRRRRRR	EEEEEEEE	
NN	NN	EEEEEEEE	TT	PPPPPPPP	RRRRRRRR	00	CC	RRRRRRRR	EEEEEEEE	
NN	NNNN	EE	TT	PP	RR	00	CC	RR	EE	
NN	NNNN	EE	TT	PP	RR	00	CC	RR	EE	
NN	NN	EE	TT	PP	RR	00	CC	RR	EE	
NN	NN	EE	TT	PP	RR	00	CC	RR	EE	
NN	NN	EEEEEEEEEE	TT	PP	RR	000000	CCCCCCCC	RR	EEEEEEEEEE	....
NN	NN	EEEEEEEEEE	TT	PP	RR	000000	CCCCCCCC	RR	EEEEEEEEEE	....

LL	IIIIII	SSSSSSSS
LL	IIIIII	SSSSSSSS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SSSSSS
LL	II	SSSSSS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SS
LLLLLLLLLL	IIIIII	SSSSSSSS
LLLLLLLLLL	IIIIII	SSSSSSSS

(2)	180	DECLARATIONS
(4)	333	NET\$PROC_XWB - Process returned XWB
(5)	544	NET\$CREATE_MBX - Create ACP mailbox
(5)	545	NET\$KILL_MBX - Delete ACP mailbox
(5)	546	NET\$MBX_QIO - Issue mailbox read
(6)	592	NET\$SET_MBX_AST - Process mailbox AST
(8)	715	NET\$CONNECT_FAIL - Notify NETDRIVER of failed link
(9)	742	NET\$SERVER_FAIL - Notify NETDRIVER of terminated server
(10)	767	NET\$SCAN_FOR_ZNA - Send pending connects to declared object
(11)	803	NET\$RESEND_SERVER - Re-send initial connect to server
(12)	836	NET\$STARTUP_OBJ - Startup privileged process
(12)	837	NET\$STARTUP_OBJ_NAM - Startup process by name
(13)	921	NET\$DELIVER_CI - Process and Deliver Inbound Connect
(14)	1048	BUILD_NCB - Build NCB for incoming connect
(15)	1125	GET_PROC - Locate process to accept connect
(16)	1361	SEND_TO_SERVER - Send connect to waiting server
(17)	1401	CREATE_SPI - Create SPI database entry
(18)	1435	GET_PR_NAM - Get name of object procedure
(18)	1436	GET_PR_ZNA - Construct ZNA string for an object
(19)	1484	TELC_DRV - Call NETDRIVER
(20)	1500	UP_CASE - Upcase the LOGINOUT strings



```
0000 1 .TITLE NETPROC - Process creation
0000 2 .IDENT 'V04-000'
0000 3 .DEFAULT DISPLACEMENT,WORD
0000 4
0000 5 *****
0000 6 *
0000 7 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 * ALL RIGHTS RESERVED.
0000 10 *
0000 11 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 * TRANSFERRED.
0000 17 *
0000 18 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 * CORPORATION.
0000 21 *
0000 22 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24 *
0000 25 *
0000 26 *****
0000 27
0000 28
0000 29
0000 30 ++
0000 31 FACILITY: NETWORK ACP
0000 32
0000 33 ABSTRACT:
0000 34 THIS MODULE PERFORMS PROCESS CREATION FOR AN INBOUND CONNECT.
0000 35
0000 36 ENVIRONMENT:
0000 37 MODE = KERNEL
0000 38
0000 39 AUTHOR: SCOTT G. DAVIS, CREATION DATE: 10-AUG-77
0000 40
0000 41 MODIFIED BY:
0000 42
0000 43 V03-024 ADE0039 Alan D. Eldridge 18-Jul-1984
0000 44 When looking for a free XWB slot, don't allow either byte of
0000 45 the local link number to be equal the character ' ' since
0000 46 that results in some non-intelligent NCB parsers to break.
0000 47
0000 48 V03-025 PRB0340 Paul Beck 18-Jul-1984 16:10
0000 49 Test against LGIS_INVPWD for invalid access instead of magic number.
0000 50
0000 51 V03-024 ADE0038 Alan D. Eldridge 25-Jun-1984
0000 52 Change SS$_NOLINKS to SS$_CONNECFail on problems finding
0000 53 or creating logical-link resources.
0000 54
0000 55 V03-023 RNG0023 Rod Gamache 12-Jun-1984
0000 56 Change calling conventions for calls to NODE COUNTER
0000 57 BLOCK access routines.
```

0000 58 :  
0000 59 :  
0000 60 :  
0000 61 :  
0000 62 :  
0000 63 :  
0000 64 :  
0000 65 :  
0000 66 :  
0000 67 :  
0000 68 :  
0000 69 :  
0000 70 :  
0000 71 :  
0000 72 :  
0000 73 :  
0000 74 :  
0000 75 :  
0000 76 :  
0000 77 :  
0000 78 :  
0000 79 :  
0000 80 :  
0000 81 :  
0000 82 :  
0000 83 :  
0000 84 :  
0000 85 :  
0000 86 :  
0000 87 :  
0000 88 :  
0000 89 :  
0000 90 :  
0000 91 :  
0000 92 :  
0000 93 :  
0000 94 :  
0000 95 :  
0000 96 :  
0000 97 :  
0000 98 :  
0000 99 :  
0000 100 :  
0000 101 :  
0000 102 :  
0000 103 :  
0000 104 :  
0000 105 :  
0000 106 :  
0000 107 :  
0000 108 :  
0000 109 :  
0000 110 :  
0000 111 :  
0000 112 :  
0000 113 :  
0000 114 :

V03-022 PRB0331 Paul Beck 1-May-1984 20:19  
1. Look for EPID instead of IPID in OBISL\_PID  
2. Fix callers of NET\$DELIVER\_CI to set up R0 correctly.

V03-021 ADE0001 Alan D. Eldridge 11-Apr-1984  
When comparing remote link addresses in NET\$PROC\_XWB, ignore  
an address of zero.

V03-019 PRB0317 Paul Beck 8-Mar-1984 17:36  
Force created network processes to use DCL as their default  
CLI, independent of the default CLI for the specified account.  
Fix bug in ADE0035.

V018 ADE0035 Alan D. Eldridge 14-Feb-1984  
Create LLI entry when receive notification of a new XWB.

V017 RNG0017 Rod Gamache 7-Feb-1984  
Fix initialization of local storage in NET\$DELIVER\_CI routine.

V016 TMH0016 Tim Halvorsen 23-Jun-1983  
Fix selection of waiting network processes so  
that processes which were activated with different  
default accounts (using default accounts on different  
objects) are correctly selected.

V015 RNG0015 Rod Gamache 20-Apr-1983  
Fix branch destinations out of range.

V014 TMH0014 Tim Halvorsen 03-Mar-1983  
If requested object name starts with a '\$', then use  
a default filespec of SYSS\$SYSTEM (rather than SYSS\$LOGIN)  
since objects with a '\$' are reserved to DEC.  
Allow STARTUP\_OBJ to be called with an object name  
as well as a number.  
Notify new DLE module of process termination.

V013 TMH0013 Tim Halvorsen 14-Feb-1983  
Remove node proxy access parameter.  
Add support for EPIDs.  
Return IPID of SPI database key in IOSB of DECLSERV QIO  
to NETSERVER process.

V012 RNG0012 Rod Gamache 26-Jan-1983  
Fix bug in NET\$DELIVER\_CI which doesn't check status  
for success on call to memory allocation routine.

V011 TMH0011 Tim Halvorsen 28-Dec-1982  
Add routine to break all links for a given process.  
Do not store NCB, SFI or PNM into SPI until the link  
is actually given to the process, and not when the  
process is created.

V010 TMH0010 Tim Halvorsen 11-Nov-1982  
Fix bug in NETSERVER startup, so that initial connects  
which have been tagged for a certain process do not get  
inadvertantly given to the another free server process



0000 115 :  
0000 116 :  
0000 117 :  
0000 118 :  
0000 119 :  
0000 120 :  
0000 121 :  
0000 122 :  
0000 123 :  
0000 124 :  
0000 125 :  
0000 126 :  
0000 127 :  
0000 128 :  
0000 129 :  
0000 130 :  
0000 131 :  
0000 132 :  
0000 133 :  
0000 134 :  
0000 135 :  
0000 136 :  
0000 137 :  
0000 138 :  
0000 139 :  
0000 140 :  
0000 141 :  
0000 142 :  
0000 143 :  
0000 144 :  
0000 145 :  
0000 146 :  
0000 147 :  
0000 148 :  
0000 149 :  
0000 150 :  
0000 151 :  
0000 152 :  
0000 153 :  
0000 154 :  
0000 155 :  
0000 156 :  
0000 157 :  
0000 158 :  
0000 159 :  
0000 160 :  
0000 161 :  
0000 162 :  
0000 163 :  
0000 164 :  
0000 165 :  
0000 166 :  
0000 167 :  
0000 168 :  
0000 169 :  
0000 170 :  
0000 171 :

for which the logical link was not intended.

V009 TMH0009 Tim Halvorsen 09-Jul-1982  
Make it possible for the network channel to be cleaned  
up without any errors.  
Add code to report mailbox messages of MSG\$\_RESET  
to the Transport module, so that it can respond to  
X.25 circuit resets during datalink startup.

V008 TMH0008 Tim Halvorsen 16-Jun-1982  
Add an entry to the SPI database when creating a network  
job, and remove it when we get the termination message.  
Add code to transfer connect requests to waiting server  
processes, in order to optimize server process creation.  
Fix code in process termination to ignore the INHIB\_MSG  
bit in the final termination status, when making the  
determination of whether the object procedure exists or not.  
Do not cause a proxy login if the connect format type is  
not a 2. This prevents an 8 byte PID from being sent to  
LOGIN for proxy logins.

V007 TMH0007 Tim Halvorsen 12-Apr-1982  
Get address of utility buffer from cell, rather than  
referencing a statically defined location.  
Modify ACP mailbox dispatching to handle X.25 mailbox  
messages, and dispatch them.  
Fix a bug in mailbox dispatching, so that if the mailbox  
read is canceled or aborted, then the QIO is re-issued.  
Make default addressing word relative and remove explicit  
addressing specifiers.

V03-06 ADE0035 A.Eldridge 11-Feb-1982  
Move check for specific OBI proxy access state to allow objects  
not in the database and with an object number zero to use the  
proxy access specified for the TASK OBI.

V03-05 ADE0034 A.Eldridge 10-Feb-1982  
Return error (instead of bug\_check) if call to \$CREMBX fails.

V03-04 ADE0033 A.Eldridge 18-Jan-1981  
Fix bug in proxy login. If the access control string received  
in the connect message is non-null then don't allow proxy  
login.

V03-03 ADE0032 A.Eldridge 26-Dec-1981  
Allow maximum task name of 12 characters in NCB.

V03-02 ADE0031 A.Eldridge 18-Dec-1981  
Make sure that the NCB, the taskname, the process name, and  
the access control strings passed to LOGINOUT, are properly  
uppercased.

V03-01 ADE0030 A.Eldridge 30-Nov-1981  
Added proxy login (access) support.

V03-00 ADE0029 A.Eldridge 01-Nov-1981  
General cleanup.

0000 172 :  
0000 173 :  
0000 174 :  
0000 175 :  
0000 176 :  
0000 177 :  
0000 178 :---

V02-17 TMH0017 Tim Halvorsen 04-Sep-1980  
Accept SYSSNET parameter as input to NET\$STARTUP\_OBJ.  
V2 A.Eldridge 01-Jan-1980  
Rewritten for Phase III

```
0000 180 .SBTTL DECLARATIONS
0000 181 :
0000 182 : INCLUDE FILES:
0000 183 :
0000 184 $ACCDDEF
0000 185 $AQBDEF
0000 186 $STSDEF
0000 187 $DRDEF
0000 188 $MSGDEF
0000 189 $PRVDEF
0000 190 $UCBDEF
0000 191 $IRPDEF
0000 192 $RCBDEF
0000 193 $LLIDEF
0000 194 $LTBDEF
0000 195
0000 196 $CNFDEF
0000 197 $CNRDEF
0000 198 $NFBDEF
0000 199 $NMADEF
0000 200 $XWBDEF
0000 201
0000 202 $WQDEF
0000 203
0000 204 $NETSYMDEF
0000 205 $NETUPDEF
0000 206 :
0000 207 : MACROS:
0000 208 :
0000 209 :
0000 210 :
0000 211 : EQUATED SYMBOLS:
0000 212 :
0000 213 :
00000024 0000 214 CNF = CNF$C_LENGTH ; Short name for readability
0000 215
00000008 0000 216 STS_M_NOACNT = 123 ; Do not generate accounting records
00000040 0000 217 STS_M_NOAUTH = 126 ; use caller's privs/quotas at login
00000080 0000 218 STS_M_NETLOG = 127 ; bit no. for network login
00000096 0000 219 MBX_MSG_LTH = 150 ; size of a mailbox message
0000000C 0000 220 MAX_TASKNAM = 12 ; Max size of taskname -- the name
0000 221 ; following teh '=' in the NCB
0000 222
0000 223 ASSUME MBX_MSG_LTH GE ACC$K_TERMLEN
```



```
0000 225 :  
0000 226 : OWN STORAGE:  
0000 227 :  
00000000 228 .PSECT NET_IMPURE,WRT,NOEXE, LONG  
0000 229  
0000 230 .ALIGN LONG  
0000 231 NET_L_R0: :  
00000000 0000 232 NET_L_FCT: .LONG 0 : Function to pass to NETDRIVER  
0004 233 NET_L_R1: :  
0004 234 NET_L_LPD: : LPD of line which is starting  
00000000 0004 235 NET_L_PID: .LONG 0 : PID to pass to NETDRIVER  
0008 236 NET_L_R2: :  
00000000 0008 237 NET_L_REASON: .LONG 0 : Disconnect reason  
000C 238 NET_L_R3: :  
00000000 000C 239 NET_L_LNK: .LONG 0 : Link number  
0010 240 NET_L_R4: :  
00000014 0010 241 NET_A_NCB: .BLKA 1 : For saving address of NCB buffer  
0014 242 NET_L_R5: :  
00000000 0014 243 NET_L_UCB: .LONG 0 : UCB address to pass to NETDRIVER  
0018 244  
00000000 0018 245 PTR_NCB_BUF: .LONG 0 : Address of NCB buffer  
00000000 001C 246 PTR_CON_BUF: .LONG 0 : Address of DELIVER_CI scratch buffer  
0020 247  
00000000 0020 248 NET_A_LLI: .LONG 0 : Address of create LLI  
00000000 00000000 0024 249 NET_Q_NCB: .QUAD 0 : NCB descriptor  
00000000 00000000 002C 250 NET_Q_PRC: .QUAD 0 : Process descriptor  
00000000 00000000 0034 251 NET_Q_TSK: .QUAD 0 : Name of file to run  
00000000 00000000 003C 252 NET_Q_ACC: .QUAD 0 : Descriptor for 3 account  
0044 253 : strings preceded by flags word  
0044 254  
00000005 0044 255 DET_C_ACC = 5 : Buffer for access control strings  
0000 0044 256 DET_AB_ACC: .WORD 0 : for creating detached, privileged  
00 0046 257 .BYTE 0 : processes. It consists of a flags  
00 0047 258 .BYTE 0 : Word followed by 3 null counted  
00 0048 259 .BYTE 0 : strings.  
0049 260  
00 0049 261 OBI_B_PRX: .BYTE 0 : OBI proxy access state  
00 004A 262 INT_B_PRX: .BYTE 0 : Internal proxy access state. This is  
004B 263 : set to "none" if any conditions are  
004B 264 : detected internally (other than the  
004B 265 : values stored in the OBI or NDI)  
004B 266 : which would disallow proxy access  
004B 267 :  
004B 268 : Fields used for termination mailbox creation, message buffering. Be  
004B 269 : careful when modifying since some code assumes data ordering without  
004B 270 : using assumes.  
004B 271 :  
004B 272 .ALIGN LONG  
0000004E 004C 273 MBX_CHAN: .BLKW 1 : Channel number of mailbox  
00000050 004E 274 MBX_RDCNT: .BLKW 1 : Number of reads posted to mailbox  
0050 275 MBX_IOSB: : I/O status block  
00000052 0050 276 .BLKW 1 : -- status of i/o completio.  
00000054 0052 277 MBX_LEN: .BLKW 1 : -- length of operation here  
00000058 0054 278 MBX_PID: .BLKL 1 : -- pid of process deleted  
0058 279 EXIT_MSG: : Buffer for mailbox message  
0000005A 0058 280 EXIT_ID: .BLKW 1 : -- message identification  
0000005C 005A 281 .BLKW 1 : -- not used
```

```
00000060 005C 282 NCB_DATA: ; On connect initiates
000000F6 005C 283 EXIT_CODE: ; status of message
000000F6 0060 284 .BLKL 1 MBX_MSG_LTH ; Leave room for message
000000F6 00F6 285 .BLKB
000000F6 00F6 286 NET$GQ_WQE_MBX:: ; MBX read element
000000F6 00F6 287 .LONG ; FLINK
000000F6 00FA 288 .LONG -4 ; BLINK
0018 00FE 289 .WORD WQE_MBX_LTH ; Length of entry
00 0100 290 .BYTE NET$C_DYN_WQE ; Structure type
05 0101 291 .BYTE WQESC-SUB-MBX ; Sub-type is 'MBX'
000000E6 0102 292 .ADDRESS MBX_ACTION ; Action routine address
00000000 0106 293 .LONG 0 ; AST parameter
00000000 010A 294 .LONG 0 ; 'In-use' flag
00000018 010E 295 WQE_MBX_LTH = .-NET$GQ_WQE_MBX
010E 296 ;
010E 297 ; Buffer to get mailbox unit number for $CREPRC argument
010E 298 ;
0000011A 010E 299 BBUF: .BLKL 3 ; Device characteristics
0000011C 011A 300 MBX_UNIT: .BLKW 1 ; Unit number for CREPRC argument
00000130 011C 301 ENDBUF: ; Truncate the rest !
011C 302 ZNABUF: .BLKB MAX_TASKNAM+8 ; Buffer for building ZNA
0130 303 ; the 8 includes 1 byte for the object
0130 304 ; number and 7 bytes of slop
0130 305
0130 306
0130 307
00000000 308 .PSECT NET_PURE,NOWRT,NOEXE, LONG
0000 309
0000 310
54 45 4E 00000008'010E0000' 0000 311 NET_Q_NETPREFIX: .ASCID 'NET' ; Prefix for unnamed tasks
00000005 000B 312 NET_Q_TASKZNA: .LONG 5 ; Length of TASK ZNA string
00000013' 000F 313 .ADDRESS TASKZNA ; Its pointer
00 0013 314 TASKZNA: .BYTE 0 ; Object type
4B 53 41 54 0014 315 .ASCII 'TASK' ; Object name
0018 316 EXIT_BUF: ; Descriptor for channel info
0000000E 0018 317 .LONG ENDBUF-BBUF ; Length of buffer
0000010E' 001C 318 .LONG BBUF ; Address of buffer
0020 319
0020 320 NET$GQ_MBX_NAME::
50 43 41 54 45 4E 00000028'010E0000' 0020 321 .ASCID 'NETACP$MBX' ; Logical name of mailbox
5B 42 4D 24 002E
0032 322 NET_Q_SYSTEM:
59 53 24 53 59 53 0000003A'010E0000' 0032 323 .ASCID 'SYS$SYSTEM:' ; Prefix for reserved objects
3A 4D 45 54 53 0040
0045 324 NET_Q_IMAGE:
59 53 24 53 59 53 0000004D'010E0000' 0045 325 .ASCID 'SYS$SYSTEM:DCL' ; Login image
4C 43 44 3A 4D 45 54 53 0053
005B 326 NET_Q_PROC:
59 53 24 53 59 53 00000063'010E0000' 005B 327 .ASCID 'SYS$SYSTEM:NETSERVER' ; Network server procedure
56 52 45 53 54 45 4E 3A 4D 45 54 53 0069
52 45 0075
0077 328 X25_DEV_NAME:
41 57 4E 0000007F'010E0000' 0077 329 .ASCID 'NWA' ; X.25 device name
0082 330
00000000 331 .PSECT NET_CODE,NOWRT, LONG
```



```
0000 333 .SBTTL NET$PROC_XWB - Process returned XWB
0000 334
0000 335
0000 336 NETDRIVER has passed us an XWB either to be linked into the LTB and assigned
0000 337 a local logical-link address (upon receiving an incoming connect) or to be
0000 338 unhooked from the LTB and deallocated.
0000 339
0000 340 If both the XWB$W_REMLNK and XWB$W_LOCLNK fields are zero, then this request
0000 341 comes from the NETACP code which handles the IOS_ACCESS request for Connect
0000 342 initiates.
0000 343
0000 344 NETACP is responsible for the LTB maintenance and the XWB linkage in order to
0000 345 avoid any race conditions it may have with NETDRIVER while scanning this list
0000 346
0000 347
0000 348 INPUTS: R3 XWB pointer
0000 349
0000 350 OUTPUTS: All registers are clobbered
0000 351
0000 352
0000 353 .SAVE PSECT
0000 354 .PSECT NET_LOCK_CODE,NOWRT,GBL ; Can't tolerate page faults
0000 355 .ENABL LSB
0000 356
0000 357 NET$PROC_XWB:: ; Process (deallocate) XWB
5B 0000'CF D0 0000 358 MOVL NET$GL_CNR_LLI,R11 ; Pick up LLI CNR
52 5A D4 0005 359 CLRL R10 ; No LLI CNF yet
55 30 A3 D0 0007 360 MOVL XWB$W_VCB(R3),R2 ; Get RCB
55 24 A2 D0 000B 361 MOVL RCB$W_PTR_LTB(R2),R5 ; Get LTB
3E A3 B5 000F 362 TSTW XWB$W_LOCLNK(R3) ; Test local link number
03 12 0012 363 BNEQ 2$ ; If NEQ, XWB being returned
0087 31 0014 364 BRW NEW_LINK ; If EQL, this is an incoming connect
0017 365 2$:
0017 366
0017 367 Locate and Delete the LLI CNF. Release hold on counter block
0017 368
0017 369
58 53 D0 0017 370 MOVL R3,R8 ; Setup XWB address for search
2E 50 E9 001A 371 $SEARCH egl,lli,l,xwb ; Find the corresponding LLI, if any
002A 372 BLBC R0,10$ ; If LBC, not found
002A 373
5E 00000064 3C BB 002A 374 PUSHR #*M<R2,R3,R4,R5> ; Save regs
56 5E D0 002C 375 SUBL #100,SP ; Create dummy non-pageable buffer
54 01 D0 0033 376 MOVL SP,R6 ; Point to dummy buffer
55 53 D0 0036 377 MOVL #1,R4 ; Say "zero XWB counters"
FFC1' 30 0039 378 MOVL R3,R5 ; XWB ptr for subr call
003C 379 BSBW NET$FLUSH_LLI_CNT ; Flush LLI and XWB counters to node
003F 380 ; counter block
5E 00000064 8F C0 003F 381 ADDL #100,SP ; Release stack space
3C BA 0046 382 POPR #*M<R2,R3,R4,R5> ; Restore regs
0048 383
58 D4 0048 384 CLRL R8 ; Nullify pointer
FFB3' 30 004A 385 BSBW CNF$PUT_FIELD ; Erase the XWB pointer
FFB0' 30 004D 386 BSBW CNF$DELETE ; Mark the entry for deletion
FFAD' 30 0050 387 BSBW CNF$PURGE ; Purge the entry from the database
5A D4 0053 388 CLRL R10 ; Forget about the LLI, its gone
FFAB' 30 0055 389 BSBW NET$RELEASE_NDCOU ; Release hold on counter block
```



```
0058 390 10$:
0058 391
0058 392
0058 393
0058 394
0058 395
50 3E A3 3C 0058 395 MOVZWL XWB$W_LOCLNK(R3),R0 ; Get link number
50 FC00 8F AA 005C 396 BICW #^C<NET$C_MAXLNK>,R0 ; Clear all but 'index' bits
50 10 A540 DE 0061 397 MOVAL LTB$SLOTS(R5)[R0],R0 ; Get link slot
53 60 D1 0066 398 CMPL (R0),R3 ; Does address match?
2F 12 0069 399 BNEQ 200$ ; If NEQ, bug
006B 400 DSBINT #NET$C_IPL ; Synchronize with NETDRIVER
0071 401
0071 402 MOVW #1,(R0)+ ; Set 'available' flag
60 80 01 B0 0074 403 MOVW XWB$W_LOCLNK(R3),(R0) ; Store last used link address
51 E0 A5 9E 0078 404 MOVAB -XWB$C_LINK - ;
007C 405 +LTB$C_LINK(R5),R1 ; Init for scan
50 51 D0 007C 406 20$: MOVL R1,R0 ; Save a copy
51 2C A1 D0 007F 407 MOVL XWB$C_LINK(R1),R1 ; Travel list
53 51 D1 0083 408 CMPL R1,R3 ; Is this it?
F4 12 0086 409 BNEQ 20$ ; If not, branch
2C A3 D0 0088 410 MOVL XWB$C_LINK(R3),- ; Remove it from list
2C A0 008B 411 XWB$C_LINK(R0)
008D 412
008D 413 ENBINT ; Restore IPL
0090 414
0090 415 DEAL_XWB: ; Deallocate XWB
50 FF6D' 30 0090 416 BSBW NET$DECR_MCOUNT ; Account for link now gone
53 D0 0093 417 MOVL R3,R0 ; Get block address for call
FF67' 30 0096 418 BSBW NET$DEALLOCATE ; Deallocate the block
05 0099 419 RSB
009A 420
009A 421 200$: BUG_CHECK NETNOSTATE,FATAL ; Else, bad slot address
009E 422
009E 423 .DSABL LSB
009E 424
009E 425
009E 426 NEW_LINK: ; Insert new XWB into LTB
009E 427
009E 428
009E 429 Find a free slot in the link table (LTB). Start from where we left
009E 430 off last time in order to avoid using the same slots over and over
009E 431 again. This technique increases the interval between re-use of a
009E 432 logical-link number -- i.e., sequence number, slot number.
009E 433
009E 434 Don't allow either byte of the local link number to equal '' since
009E 435 some non-intelligent NCB parsers mistake that for the end of the
009E 436 NCB.
009E 437
009E 438 The slot vector terminates with a -1 (longword) followed by a
009E 439 0 (longword).
009E 440
009E 441
009E 442
50 0000'8F 3C 009E 442 MOVZWL #SS$ CONNECFAIL,R0 ; Assume failure
54 65 D0 00A3 443 MOVL LTB$C_SLT_NXT(R5),R4 ; Get first slot candidate ptr
FD 84 E9 00A6 444 5$: BLBC (R4)+,5$ ; LBC means unavailable
FFFFFFFF 8F 74 D1 00A9 445 CMPL -(R4),#-1 ; Backup and test for end of
22 12 00B0 446 BNEQ 10$ ; NEQ means slot found
```

```
54 10 A5 DE 00B2 447 MOVAL LTB$SLOTS(R5),R4 : Start from top of vector
      FD 84 E9 00B6 448 7$: BLBC (R4)+,7$ : LBC means unavailable
FFFFFFFF 8F 74 D1 00B9 449 CMPL -(R4),#-1 : Backup and test for end of
      66 13 00C0 450 BEQL 200$ : EQL means slot not found
02 A4 0400 8F A0 00C2 451 8$: ADDW #NET$C_MAXLNK+1,2(R4) : Update local link seq #
      22 02 A4 91 00C8 452 CMPB 2(R4),#^A''' : Is low byte a double quote?
      E8 13 00CC 453 BEQL 7$ : If EQL yes, keep scanning
      22 03 A4 91 00CE 454 CMPB 3(R4),#^A''' : Is high byte a double quote ?
      EE 13 00D2 455 BEQL 8$ : If EQL, bump the seq # and try again
      00D4 456 10$:
      00D4 457
      00D4 458
      00D4 459
      00D4 460
      00D4 461
      00D4 462
      00D4 463
      00D4 464
50 E0 A5 9E 00D4 464 MOVAB -XWB$S_LINK - : Init for scan
      51 50 D0 00D8 465 +LTB$S_XWB(R5),R0 : Remember last entry
50 2C A0 D0 00DB 466 30$: MOVL R0,R1 : Go to next entry
      15 13 00DF 467 MOVL XWB$S_LINK(R0),R0 : If EQL, at end of list
      3A A3 B1 00E1 468 BEQL 50$ : Are we going too far ?
      3A A0 00E4 469 CMPW XWB$S_REMNOD(R3),- :
      0E 1A 00E6 471 50$ : If GTRU yes, stop here
      3C A3 B1 00E8 472 BGTRU 50$ : Is this it ?
      3C A0 00EB 473 CMPW XWB$S_REMLNK(R3),- :
      E9 12 00ED 474 30$ : If NEQ no, continue searching
      3C A3 B5 00EF 475 TSTW XWB$S_REMLNK(R3) : But, if =0 then no address has been
      E4 13 00F2 476 BEQL 30$ : assigned; comparison was invalid
      9A 11 00F4 477 BRB DEAL_XWB : ...else duplicate connect
      00F6 478 50$:
      00F6 479
      00F6 480
      00F6 481
      00F6 482
      00F6 483
      00F6 484
      00FC 485
      00FC 486
3E A3 0E A3 AA 00FC 486 BICW #XWB$M_STS SOL,- :
      02 A4 B0 00FE 487 XWB$S_STS(R3) : No longer queued
      84 53 D0 0100 488 MOVW 2(R4),XWB$S_LOCLNK(R3) : Setup local link number
      65 54 D0 0105 489 MOVL R3,(R4)+ : Store XWB ptr in this slot
      2C A3 50 D0 0108 490 MOVL R4,LTB$S_SLT NXT(R5) : Store scan's next starting pt.
      2C A1 53 D0 010B 491 MOVL R0,XWB$S_LINK(R3) : Link tail of list to current XWB
      0113 492 MOVL R3,XWB$S_LINK(R1) : Link XWB to head of list
      0113 493
      0113 494
      0116 495
      FEE7' 30 0116 496 ENBINT : Restore IPL
      0119 497 BSBW CREATE_LLI : Create LLI and insert it into database
      3C A3 B5 0119 498 TSTW XWB$S_REMLNK(R3) : Use status as input to NET$DELIVER_CI
      09 13 011C 499 BEQL 100$ : Connect Initiate ?
      56 53 D0 011E 500 MOVL R3,R6 : If EQL yes, return R0 to caller
      02B8' 30 0121 501 BSBW NET$DELIVER_CI : Else, copy XWB address
      0124 502 : Create LLI, and deliver connect
      50 01 D0 0124 503 MOVL #1,R0 : notification to some server
      : Say "success"
```

```
05 0127 504 100$: RSB ; Done
    0128 505
    0128 506 200$: BUG_CHECK NETNOSTATE,FATAL
    012C 507
00000000 508 .RESTORE_PSECT
    0000 509
    0000 510
    0000 511 CREATE_LLI: ; Create LLI and insert it into the list
    0000 512
    0000 513 This subroutine is required so that the 'utility buffer' acquired
    0000 514 by the NET$GETUTLBUF co-routine will be released in a timely manner.
    0000 515
    0000 516 NOTE - the NET$ACQUIRE_NDCOU routine needs the utility buffer, so
    0000 517 we must not allocate the utility buffer until after we acquire the
    0000 518 NDC counter block.
    0000 519
    FFFD' 30 0000 520 BSBW NET$ACQUIRE_NDCOU ; Inc. reference level of counter block
    40 50 E9 0003 521 BLBC R0,90$ ; If LBC, problem encountered
    FFF7' 30 0006 522 BSBW NET$GETUTLBUF ; Get permission to use utility buffer
    0009 523 ; - the above is a co-routine call
    5B 0000'CF D0 0009 524 MOVL NET$GL_CNR_LLI,R11 ; Pick up CNR
    FFEF' 30 000E 525 BSBW CNF$INIT_UTL ; Init 'utility buffer' as a CNF
    5B 53 D0 0011 526 MOVL R3,R8 ; Get XWB
    0014 527 $PUTFLD LLI,L,xwb ; ...Store it in LLI
    001F 528
    007C 8F BB 001F 529 PUSHR #M<R2,R3,R4,R5,R6> ; Save registers
    56 2C AA 9E 0023 530 MOVAB CNF+LLI$Z_NDC RI(R10),R6 ; Point to 'running total' counters
    66 1C 00 56 6E 00 2C 0027 531 MOVCS #0,(SP),#0,#NDC$C_LENGTH,(R6) ; Zero the counters
    56 48 AA 9E 002D 532 MOVAB CNF+LLI$Z_NDC LZ(R10),R6 ; Point to 'last zeroed' counters
    66 1C 00 56 6E 00 2C 0031 533 MOVCS #0,(SP),#0,#NDC$C_LENGTH,(R6) ; Zero the counters
    56 D4 0037 534 CLRL R6 ; No 'old' CNF
    FFC4' 30 0039 535 BSBW CNF$INSERT ; Try to put block into list
    007C 8F BA 003C 536 POPR #M<R2,R3,R4,R5,R6> ; Restore registers
    0040 537
    0B 50 E8 0040 538 BLBS R0,100$ ; If LBS, okay
    FFBA' 30 0043 539 BSBW NET$RELEASE_NDCOU ; Else, dec. reference to counter block
    50 0000'8F 3C 0046 540 90$: MOVZWL #SS$_CONNECFail,R0 ; Return general purpose error status
    05 004B 541 100$: RSB ; Release utility buffer, return status
    004C 542
```



```
004C 544 .SBTTL NET$CREATE_MBX - Create ACP mailbox
004C 545 .SBTTL NET$KILL_MBX - Delete ACP mailbox
004C 546 .SBTTL NET$MBX_QIO - Issue mailbox read
004C 547 :++
004C 548 :
004C 549 :
004C 550 : *** TBS ***
004C 551 :
004C 552 :--
004C 553 NET$CREATE_MBX::
004C 554 CLRW MBX_RDCNT ; Init outstanding mailbox read count
0050 555 $CREMBX S - ; Create mailbox
0050 556 CHAR = MBX_CHAN,-
0050 557 MAXMSG = #MBX_MSG_LTH,-
0050 558 BUFQUO = #<MBX_MSG_LTH*16>,-
0050 559 LOGNAM = NET$QIO_MBX_NAME,- ; mailbox's logical name
0050 560 PROMSK = #0
16 50 E9 0071 561 BLBC R0,10$ ; Br if error
0074 562 $GETCHN S - ; Get mailbox unit number
0074 563 CHAR = MBX_CHAN,-
0074 564 PRIBUF = EXIT_BUF
05 008A 565 10$: RSB ; Return status in R0
008B 566
008B 567
008B 568 NET$KILL_MBX::
008B 569 $DASSGN_S CHAN = MBX_CHAN ; Delete channel to mailbox
008B 570 RSB ; do it
05 0097 571
0098 572
0098 573
0098 574 NET$MBX_QIO:: ; Post read to mailbox
0098 575 :
0098 576 :
0098 577 : This routine puts a read out on the mailbox for process termination and
0098 578 : inbound connect notifications.
0098 579 :
0098 580 :
0098 581 $QIO_S CHAN = MBX_CHAN,-
0098 582 FUNC = S^#IOS_READVBLK,-
0098 583 EFN = #NET$C-EFN_ASYN,-
0098 584 ASTADR = NET$SET_MBX_AST,-
0098 585 IOSB = MBX_IOSB,-
0098 586 P1 = EXIT_MSG,-
0098 587 P2 = #MBX_MSG_LTH
04 50 E8 00C1 588 BLBS R0,10$ ; Br unless error
00C4 589 BUG_CHECK ACPMBFAIL,FATAL ; !arrgh
05 00C8 590 10$: RSB ; return
```

```
00C9 592 .SBTTL NET$SET_MBX_AST - Process mailbox AST
00C9 593 :++
00C9 594 :
00C9 595 :
00C9 596 :--
00C9 597 NET$SET_MBX_AST::
003C 00C9 598 .WORD ^M<R2,R3,R4,R5>
00CB 599
50 00F6'CF 9E 00CB 600 MOVAB NET$GQ_WQE_MBX,R0 ; Get base of mailbox WQE
14 A0 D5 00D0 601 TSTL WQE$S_PM2(R0) ; Is it active?
OD 12 00D3 602 BNEQ 10$ ; If NEQ then active, there's a bug
10 A0 04 AC D0 00D5 603 MOVL 4(AP),WQE$S_PM1(R0) ; Get the AST parameter
14 A0 01 CE 00DA 604 MNEGL #1,WQE$S_PM2(R0) ; Mark WQE busy
FF1F' 30 00DE 605 BSBW WQE$INSQOE ; Queue the WQE
04 00E1 606 RET ; Done
00E2 607
00E2 608 10$: BUG_CHECK NETNOSTATE,FATAL ; Signal the bug
00E6 609
00E6 610 MBX_ACTION: ; Enter upon WQE dispatch
00E6 611 CLRL WQE$S_PM2(R5) ; Mark WQE idle
EE'AF 14 A5 D4 00E9 612 CALLS #0,B^NET$MBX_AST ; Call the mailbox processor
00 00 FB 05 00ED 613 RSB
00EE 614 :+
00EE 615 :
00EE 616 NET$MBX_AST - THIS ROUTINE SERVICES PROCESS TERMINATIONS
00EE 617 AND INBOUND CONNECT NOTIFICATIONS
00EE 618 :
00EE 619 :--
0000 00EE 620 NET$MBX_AST:: .WORD 0 ; Entry point
00F0 621 CMPW MBX_IOSB,S^#SS$ _ABORT ; Was the i/o cancelled?
00F5 622 BEQL 5$ ; If so, assume mailbox going away
0000'8F 0050'CF B1 00F7 623 CMPW MBX_IOSB,#SS$ _CANCEL ; Try this code, too
05 13 00FE 624 BEQL 5$ ; If NEQ proceed
04 10 0100 625 BSBB 10$ ; Dispatch
FF93 30 0102 626 BSBW NET$MBX_QIO ; Put out another read
04 0105 627 5$: RET ; Done
0106 628 :
0106 629 : Dispatch
0106 630 :
50 0054'CF D0 0106 631 10$: MOVL MBX_PID,R0 ; Get EPID returned by MBX driver
00000000'GF 16 010B 632 JSB G^EXE$EPID_TO_IPID ; Convert to internal PID
0054'CF 50 D0 0111 633 MOVL R0,MBX_PID ; Use the IPID for later processing
56 0000'CF D0 0116 634 MCVL NET$GQ_NET_UCB,R6 ; Point to our NET channel's UCB
5B 0058'CF 9E 011B 635 MOVAB EXIT_ID,R1T ; Get address of mbx message
56 8B B0 0120 636 MOVW (R11)+,R6 ; Get message type
59 8B B0 0123 637 MOVW (R11)+,R9 ; Get unit number
5A 8B 9A 0126 638 MOVZBL (R11)+,R10 ; Get device name count value
00 6B 5A 2D 0129 639 CMPC5 R10,(R11),#0,- ; X.25 mailbox message?
007B'DF 0077'CF 012D 640 X25_DEV_NAME,@X25_DEV_NAME+4
5B 5A C0 0133 641 BEQL 20$ ; Branch if so
0135 642 ADDL R10,R11 ; Get pointer to mbx "data"
0138 643 $DISPATCH TYPE=W,R6,- ; Dispatch on mailbox msg type
0138 644 <-
0138 645 <MSG$ _DELPROC, DELPROC>,- ; Process termination
0138 646 <MSG$ _CONNECT, CONNECT>,- ; Inbound connect
0138 647 <MSG$ _PATHLOST, NET$DRV _CANCEL>,- ; I/O channel cancelled
0138 648 >
```

```
05 01A4 649 RSB ; Ignore the message
    01A5 650
    01A5 651
    01A5 652 ; Dispatch on X.25 mailbox message
    01A5 653
    01A5 654 20$: MOVAB EXIT_ID+20,R11 ; Point to 'data'
SA 0052'CF 3C 01AA 655 MOVZWL MBX_LEN,R10 ; Get length of mailbox message
SA 14 C2 01AF 656 SUBL #20,R10 ; Subtract out overhead
    01B2 657 $DISPATCH TYPE=W,R6,- ; Dispatch on mailbox msg type
    01B2 658 <- ; Incoming X.25 call
    01B2 659 <MSG$_CONNECT, NET$DLL_X25_CALL>,- ; X.25 circuit reset
    01B2 660 <MSG$_RESET, NET$DLL_X25_RESET>,-
    01B2 661 >
05 01D6 662 RSB ; Ignore the message
```



```
01D7 664 :
01D7 665 : Connect initiate message received
01D7 666 :
01D7 667 :
01D7 668 : The mailbox data consists of the address of the NETDRIVER update
01D7 669 : routine and the address of the XWB.
01D7 670 :
01D7 671 : CONNECT:
56 04 AB D0 01D7 672 : MOVL 4(R11),R6 : Get XWB address
50 01 D0 01DB 673 : MOVL #1,R0 : "Success" flag to NET$DELIVER_CI
01FB 30 01DE 674 : BSBW NET$DELIVER_CI : Deliver the inbound connect to a user
05 01E1 675 : RSB
01E2 676 :
01E2 677 :
01E2 678 : Handle network process termination
01E2 679 :
01E2 680 : DELPROC:
01E2 681 :
01E2 682 : Notify netdriver of process exit
01E2 683 :
52 04 D0 01E2 684 : MOVL #NET$C_DR_NOBJ,R2 : Assume can't find .com file
005C'CF 00000000'8F D1 01E5 685 : CLRBIT #ST$SV_INHIB_MSG,EXIT_CODE : Ignore INHIB_MSG flag
11 13 01F4 686 : CMPL #RMS$_FNF,EXIT_CODE : file-not-found?
52 26 D0 01F6 687 : BEQL 35$ : if EQL yes
005C'CF 00000000'8F D1 01F9 688 : MOVL #NET$C_DR_EXIT,R2 : assume just an exit
03 12 0202 689 : CMPL #LGIS_INVPWD,EXIT_CODE : was it an access problem?
52 22 D0 0204 690 : BNEQ 35$ : if NEQ, no
51 0054'CF D0 0207 691 : MOVL #NET$C_DR_ACCESS,R2 : say so
005E 30 020C 692 35$: MOVL MBX_PID,RT : PID of exiting process
020F 693 : BSBW NET$SERVER_FAIL : Notify NETSERVER server gone
020F 694 :
020F 695 : Remove the process from the SPI database
020F 696 :
5B 0000'CF D0 020F 697 : MOVL NET$GL_CNR_SPI,R11 : Get root of SPI database
5A D4 0214 698 : CLRL R10 : Start at beginning of list
58 0054'CF D0 0216 699 : MOVL MBX_PID,R8 : Get process PID
18 50 E9 0218 700 : $SEARCH egl_spi,l,pid : Find database entry
04 50 E9 0228 701 : BLBC R0,50$ : Branch if not found
022B 702 : $GETFLD spi,l,irp : Waiting DECLSERV IRP?
0236 703 : BLBC R0,40$ : Branch if no IRP waiting
0239 704 : BUG_CHECK NETNOSTATE,FATAL : Should never have IRP
FDC0' 30 023D 705 40$: BSBW CNF$DELETE : waiting, if process deleted
FDBD' 30 0240 707 50$: BSBW CNF$PURGE : Mark the CNF entry deleted
0243 708 : Delete all marked CNFs
0243 709 :
0243 710 : Notify DLE module of process termination.
58 0054'CF D0 0243 711 : MOVL MBX_PID,R8 : Setup the PID
FDB5' 30 0248 712 : BSBW DLE$PRC_EXIT : Inform DLE of process exit
05 024B 713 : RSB
```

```
024C 715 .SBTTL NET$CONNECT_FAIL - Notify NETDRIVER of failed link
024C 716 :+
024C 717 :
024C 718 : An attempt to confirm a logical link has failed. Notify NETDRIVER so that
024C 719 : it can verify the user's access to the link and then notify the remote end of
024C 720 : the link that the link is being broken and why.
024C 721 :
024C 722 :
024C 723 : INPUTS:      R3      Local logical link number (0 implies connect initiate)
024C 724 :           R2      Reason code to be sent in the disconnect message
024C 725 :           R1      User's PID
024C 726 :
024C 727 : OUTPUTS:     R5-R0   Clobbered
024C 728 :
024C 729 :           All other registers are preserved
024C 730 :
024C 731 :-
024C 732 NET$CONNECT_FAIL::
0014'CF 0000'CF D0 024C 733      MOVE    NET$GL_NET_UCB,NET_L_UCB      ; A connect attempt has failed
000C'CF 53      3C 0253 734      MOVZWL  R3,NET_L_LNK          ; Use the ACP's UCB
12      13 0258 735      BEQL     10$          ; Specify link number
0008'CF 52      D0 025A 736      MOVL    R2,NET_L_REASON        ; If EQL then connect initiate
0004'CF 51      D0 025F 737      MOVL    R1,NET_L_PID          ; Specify disconnect reason
0000'CF 01      D0 0264 738      MOVL    #NETUPD$-ABORT,NET_L_FCT ; Specify user's PID
067F    30 0269 739      MOVL    #NETUPD$-ABORT,NET_L_FCT ; Specify "link terminated"
05 026C 740 10$:  RSB      TELL_DRV      ; Notify NETDRIVER
```

```
026D 742 .SBTTL NET$SERVER_FAIL - Notify NETDRIVER of terminated server
026D 743 :+
026D 744 :
026D 745 A server process (or NETSERVER session) has terminated. Notify NETDRIVER so
026D 746 that it can break all links that might still be pending for that process.
026D 747 This handles the case where the process was unable to confirm the link due
026D 748 to some error.
026D 749 :
026D 750 :
026D 751 INPUTS: R2 Reason code to be sent in the disconnect message
026D 752 R1 User's PID
026D 753 :
026D 754 OUTPUTS: R5-R0 Clobbered
026D 755 :
026D 756 ALL other registers are preserved
026D 757 :
026D 758 :-
026D 759 NET$SERVER_FAIL::
0014'CF 0000'CF D0 026D 760 MOVL NET$GL_NET_UCB,NET_L_UCB ; A server has terminated
0008'CF 52 D0 0274 761 MOVL R2,NET_L_REASON ; Use the ACP's UCB
0004'CF 51 D0 0279 762 MOVL R1,NET_L_PID ; Specify disconnect reason
0000'CF 03 D0 027E 763 MOVL #NETUPDS_EXIT,NET_L_FCT ; Specify user's PID
0665 30 0283 764 BSBW TELL_DRV ; Specify "process exit"
05 0286 765 RSB ; Notify NETDRIVER
```



NET\$SCAN\_FOR\_ZNA - Send pending connects

0287 767 .SBTTL NET\$SCAN\_FOR\_ZNA - Send pending connects to declared object

0287 768

0287 769

0287 770

0287 771

0287 772

0287 773

0287 774

0287 775

0287 776

0287 777

0287 778

0287 779

0287 780

0287 781

0287 782

0287 783

0287 784

0287 785

0287 786

0287 787

0287 788

0287 789

0287 790

0287 791

0287 792

0287 793

0287 794

0287 795

0287 796

0287 797

0287 798

0287 799

0287 800

0287 801

0287 802

0287 803

0287 804

0287 805

0287 806

0287 807

0287 808

0287 809

0287 810

0287 811

0287 812

0287 813

0287 814

0287 815

0287 816

0287 817

0287 818

0287 819

0287 820

0287 821

0287 822

0287 823

0287 824

0287 825

0287 826

0287 827

0287 828

0287 829

0287 830

0287 831

0287 832

0287 833

0287 834

0287 835

0287 836

This routine is called when a task name or object is declared by a user.  
The function is to scan the XWB list for links in the Connect Initiate  
state which are intended for the object with the specified ZNA and to build  
a NCB which is given to NETDRIVER to be put in the declared name's mailbox.

INPUTS: R7,R8 = Descriptor of object ZNA being declared

OUTPUTS: None

NET\$SCAN\_FOR\_ZNA::

MOVQ R7,R9

MOVQ NET\$GL\_PTR,VCB,R6

MOVQ RCB\$PTR,LTB(R6),R6

MOVAB LTB\$SLOTS+4(R6),R5

10\$: BLBS (R5)+10\$

MOVQ -4(R5),R6

BEQ 30\$

CMPB #XWB\$C\_STA,CIR,-

XWB\$B\_STA(R6)

BNEQ 10\$

MOVAB XWB\$T\_LPRNAM(R6),R1

PUSHR #M&lt;R5,R6,R9,R10&gt;

BSBW GET\_PR\_ZNA

BLBC R0,20\$

CMPC5 R7,(R8),#0,R9,(R10)

BNEQ 20\$

MOVQ #1,R0

BSBW NET\$DELIVER\_CI

20\$: POPR #M&lt;R5,R6,R9,R10&gt;

BRB 10\$

RSB

; Find unclaimed XWBs which match ZNA  
; Make copy of ZNA descriptor  
; Get RCB pointer  
; Get LTB pointer  
; Point to first XWB (skip slot 0)  
; If LBS then pointer is invalid  
; Get the XWB address  
; If EQL then done  
; In connect initiate state?  
; If NEQ then keep looking  
; Setup for subroutine call  
; Save important regs  
; Get ZNA string from LPRNAM  
; If LBC then field is not valid  
; Are they the same?  
; If NEQ keep looking  
; "Success" flag to NET\$DELIVER\_CI  
; Build NCB, pass to user in mailbox  
; Restore regs  
; Keep looking  
; Done

59 57 7D  
56 0000 CF D0  
56 24 A6 D0  
55 14 A6 9E  
56 FD 85 E8  
56 FC A5 D0  
29 13  
03 91  
1E A6  
F1 12  
51 00A5 C6 9E  
0660 8F BB  
05F1 30  
0E 50 E9  
6A 59 00 68 57 2D  
06 12  
50 01 D0  
0119 30  
0660 8F BA  
CE 11  
05 02C9

```
02CA 803 .SBTTL NET$RESEND_SERVER - Re-send initial connect to server
02CA 804 :+
02CA 805 :
02CA 806 : This routine is called when a server process declares that it is waiting
02CA 807 : for an incoming connect. The XWB list is scanned for links in the CI
02CA 808 : state looking to see if the initial connect which started the process
02CA 809 : is still pending. If so, then re-send the NCB to the server process
02CA 810 : so that it will be executed.
02CA 811 :
02CA 812 : INPUTS:      R8 = PID of server process
02CA 813 :
02CA 814 : OUTPUTS:     None
02CA 815 :
02CA 816 :-
02CA 817 NET$RESEND_SERVER::
56 0000'CF D0 02CA 818      MOVL    NET$GL_PTR_VCB,R6      ; Find unclaimed XWBs for server process
56 24 A6 D0 02CF 819      MOVL    RCB$PTR_LTB(R6),R6      ; Get RCB pointer
55 14 A6 9E 02D3 820      MOVAB   LTB$SLOTS+4(R6),R5      ; Get LTB pointer
56 FC A5 D0 02DA 821 10$:    BLBS    (R5)+-10$          ; Point to first XWB (skip slot 0)
1C 13 02DE 822      MOVL    -4(R5),R6          ; If LBS then pointer is invalid
03 91 02E0 823      BEQL     30$              ; Get the XWB address
1E A6 02E2 824      CMPB     #XWB$C_STA_CIR,-      ; If EQL then done
F1 12 02E4 825      XWB$B_STA(R6)              ; In connect initiate state?
34 A6 58 D1 02E6 826      BNEQ    10$          ; If NEQ then keep looking
EB 12 02EA 827      CMPL     R8,XWB$B_PID(R6)      ; Intended for this process?
0160 8F BB 02EC 828      BNEQ    10$          ; If NEQ keep looking
50 01 D0 02F0 829      PUSHR   #^M<R5,R6,R8>      ; Save registers
00E6 30 02F3 830      MOVL     #1,R0              ; "Success" flag to NET$DELIVER_CI
0160 8F BA 02F6 831      BSBW    NET$DELIVER_CI    ; Build NCB, satisfy DECLSERV request
DB 11 02FA 832      POPR     #^M<R5,R6,R8>      ; Restore registers
05 02FC 833      BRB        10$              ; Keep looking
834 30$:    RSB        10$              ; Done
```

```
02FD 836 .SBTTL NET$STARTUP_OBJ - Startup privileged process
02FD 837 .SBTTL NET$STARTUP_OBJ_NAM - Startup process by name
02FD 838 +
02FD 839
02FD 840 Startup a privileged object process if it is not already running. This is
02FD 841 used to create EVL for event logging and NML for down-line loading or
02FD 842 up-line dumping.
02FD 843
02FD 844 Inputs:
02FD 845
02FD 846 R8 = Object number to start (If NET$STARTUP_OBJ)
02FD 847 R7/R8 = Object name to start (If NET$STARTUP_OBJ_NAM)
02FD 848
02FD 849 R2,R3 = Descriptor of string to be passed as SYS$NET to process
02FD 850 R4,R5 = Descriptor of string to be used as process name
02FD 851 If =0 then use the object's name as the process name
02FD 852
02FD 853 Outputs:
02FD 854
02FD 855 R1 PID if process has been created
02FD 856 R0 Status
02FD 857 -
02FD 858 .ENABL LSB
02FD 859
02FD 860 NET$STARTUP_OBJ_NAM::
02FD 861 PUSHRR #M<R7,R8,R9,R10,R11> ; Save registers
5B 0F80 8F BB 02FD 862 MOVL NET$GL_CNR_OBI,R11 ; Point to OBI database
0000'CF D0 0301 863 CLRL R10 ; and start at beginning of list
5A D4 0306 864 $SEARCH egl,obi,l,nam ; Search for specified object
18 11 0315 865 BRB 1$ ; Join common code
0317 866
02FD 867 NET$STARTUP_OBJ::
02FD 868 PUSHRR #M<R7,R8,R9,R10,R11> ; Startup privileged process
5B 0F80 8F BB 02FD 869 MOVL NET$GL_CNR_OBI,R11 ; Save registers
0000'CF D0 031B 870 CLRL R10 ; Point to OBI database
5A D4 0320 871 $SEARCH egl,obi,l,num ; and start at beginning of list
0322 872 1$: CLRL R1 ; Search for specified object
0024'CF 51 D4 032F 873 MOVQ R2,NET_Q_NCB ; Clear PID
1C 50 E9 0336 874 BLBC R0,2$ ; Store descriptor of SYS$NET string
0339 875 ; Skip if not defined as object
0339 876 ; If object has already declared itself, then it is running
0339 877
02FD 878 $GETFLD obi,l,ucb ; If UCB NE 0, it has declared itself
OE 50 E8 0344 879 BLBS R0,2$ ; If declared, then its already running
0347 880
02FD 881 ; If not, get the access control string and process name
0347 882
02FD 883 $GETFLD obi,s,sfi ; Get the process file name
03 50 E8 0352 884 BLBS R0,5$ ; Skip if specified
007F 31 0355 885 BRW 80$ ; Return with status in R0
0034'CF 57 7D 0358 886 2$: MOVQ R7,NET_Q_TSK ; Save the descriptor
002C'CF 54 7D 035D 887 MOVQ R4,NET_Q_PRC ; Setup process name
10 12 0362 888 BNEQ 10$ ; If NEQ then name is non-null
0364 889 $GETFLD obi,s,nam ; Else get object name
002C'CF 57 7D 036F 890 MOVQ R7,NET_Q_PRC ; Use as process name
003C'CF 05 D0 0374 891 10$: MOVL #DET_C_ACC,NET_Q_ACC ; Setup descriptor of access control
0040'CF 0044'CF 9E 0379 892 MOVAB DET_XB_ACC,NET_Q_ACC+4 ; data used for create detached,
```



```
0380 893
0380 894
0380 895
0380 896
0380 897
0380 898
0380 899
0380 900
0380 901
0380 902
0380 903
0380 904
0380 905
0380 906
0380 907
0380 908
0380 909
0380 910
0380 911
0380 912
0380 913
0380 914
0380 915
0380 916
0380 917
0380 918
0380 919

18 50
50 0004'CF
00000000'GF
0004'CF 50
51 0004'CF
50 00'
0F80 8F

E9
D0
16
D0
D0
D0
BA
05
03DC
03DC

80$:

BLBC RO,80$
MOVL NET_L_PID,R0
JSB G^EXE$EPID TO_IPID
MOVL R0,NET_L_PID
MOVL NET_L_PID,R1
MOVL S^#SS$_NORMAL,R0
POPR #^M<R7,R8,R9,R10,R11>
RSB

.DSABL LSB

; privileged processes.
; Start the process with privileges
$CREPRC 5
INPUT = NET_Q_TSK,-
OUTPUT = NET_Q_ACC,-
ERROR = NET_Q_NCB,-
PRCNAM = NET_Q_PRC,-
IMAGE = NET_Q_IMAGE,-
PIDADR = NET_L_PID,-
MBXUNT = MBX_UNIT,-
BASPRI = G^SYSS$GB_DEFPR1,-
UIC = #<^01@167^04>,-
STSFLG = <#STS_M_NETLOG!-
STS_M_NOAUTH!-
STS_M_NOACNT>

; create a process
; Network .COM filename
; Network access control string
; SYSSNET logical name string
; Process name
; Image (LOGINOUT) to run first
; Place to store process id
; MBX for termination
; Priority
; UIC is [1,4]
; Network login parameters (IN,OUT,ERR)
; Use caller's privs/quotas/etc.
; Do not add any accounting records
; If LBC then failed
; Get the EPID returned by CREPRC
; Convert to internal PID format
; Use internal format of PID
; Return the PID to caller
; Success
; Restore registers
```

```
03DC 921 .SBTTL NET$DELIVER_CI - Process and Deliver Inbound Connect
03DC 922 ++
03DC 923
03DC 924 A non-zero destination object number indicates that NETACP must fetch the
03DC 925 name of the .COM file from the OBJ block - using 'SYS$SYSROOT:[SYSEXE]' as
03DC 926 the default directory. A zero destination object number indicates that the
03DC 927 .COM file name is the same as the destination taskname - the default login
03DC 928 directory account is assumed to contain the taskname.COM.
03DC 929
03DC 930 \update this to include tasks with a file i.d.\;!
03DC 931
03DC 932 If the incoming USER,PSW,ACCT strings are all null, then the default
03DC 933 inbound access control for the specified object (or task) are used (these
03DC 934 strings may also be null). This allows a DECnet-VAX node to serve as a
03DC 935 convenient host particularly for RSX-11S.
03DC 936
03DC 937 This routines determines whether the connect is to be handed to a task
03DC 938 which has declared a name or an object type.
03DC 939
03DC 940 INPUTS: R11 LLI CNR address (if low bit set in R0)
03DC 941 R10 LLI CNF address (if low bit set in R0)
03DC 942 R6 XWB address
03DC 943 R0 Low bit set => deliver connect notification
03DC 944 Low bit clear => tell NETDRIVER that resource error
03DC 945 occurred
03DC 946
03DC 947 OUTPUTS: R11,R10,R6 are preserved.
03DC 948
03DC 949 All other registers are clobbered.
03DC 950
03DC 951 SIDE EFFECTS: Process created if needed, image started
03DC 952
03DC 953 --
03DC 954
03DC 955 Define scratch storage
03DC 956
03DC 957
03DC 958
03DC 959
03DC 960
0000000C 03DC 961 ACC = 12 ; Composite access strings
000000C8 03DC 962 PRC = 200 ; Process name
0000012C 03DC 963 TSK = 300 ; Image to run
000003E8 03DC 964 CONN_SPACE = 1000 ; Size of scratch storage
03DC 965
03DC 966 NET$DELIVER_CI:
0018'CF D4 03DC 967 CLRC PTR_NCB_BUF ; No NCB buffer yet
001C'CF D4 03E0 968 CLRL PTR_CON_BUF ; No scratch buffer yet
03E4 969
03E4 970 Initialize parameters for call to NETDRIVER
03E4 971
03E4 972 MOVZWL XWBSW LOCLNK(R6),NET_L_LNK ; Setup logical link address
000C'CF 3E A6 3C 03E4 973 MOVZBL #NETUPDS_ABORT, NET_L_FCT ; Assume process couldn't start
0000'CF 01 9A 03EA 974 MOVL NET$GL_NET_UCB, NET_L_UCB ; Default is our UCB
0014'CF 0000'CF D0 03EF 975 MOVL R10, NET_A_LLI ; Save LLI pointer
0020'CF SA D0 03F6 976 CLRL NET_L_PID ; No PID yet
0004'CF D4 03FB 977 CLRL NET_A_NCB ; No NCB yet
0010'CF D4 03FF
```

			0403	978	:		
			0403	979	:		
			0403	980	:		
	57 50	E9	0403	981	BLBC	R0,3\$	: If LBC, resource error encountered
			0406	982			: by caller
51	03E8 8F	3C	0406	983	MOVZWL	#CONN_SPACE,R1	: Set size of scratch buffer
	FBF2'	30	040B	984	BSBW	NET\$ALLOCATE	: Allocate a scratch buffer
	4C 50	E9	040E	985	BLBC	R0,3\$	: Br if allocation failure, notify
			0411	986			: driver
001C'CF	52	D0	0411	987	MOVL	R2,PTR_CON_BUF	: Save address for deallocation
			0416	988	:		
			0416	989	:	Initialize descriptors and data for process creation	
			0416	990	:		
53	0C A2	9E	0416	991	MOVAB	ACC(R2),R3	: Get ACC address
0040'CF	53	D0	041A	992	MOVL	R3,NET_Q_ACC+4	: Store it
003C'CF	53	CE	041F	993	MNEGL	R3,NET_Q-ACC	: Bias ACC size
53	00C8 C2	9E	0424	994	MOVAB	PRC(R2),R3	: Get PRC address
0030'CF	53	D0	0429	995	MOVL	R3,NET_Q_PRC+4	: Store it
002C'CF	53	CE	042E	996	MNEGL	R3,NET_Q-PRC	: Bias PRC size
53	012C C2	9E	0433	997	MOVAB	TSK(R2),R3	: Get TSK address
0038'CF	53	D0	0438	998	MOVL	R3,NET_Q_TSK+4	: Store it
0034'CF	53	CE	043D	999	MNEGL	R3,NET_Q-TSK	: Bias TSK size
			0442	1000	:		
			0442	1001	:	Set default values	
			0442	1002	:		
50	0000'CF	D0	0442	1003	MOVL	NET\$GL_PTR_VCB,R0	: Point to RCB
0049'CF	67 A0	90	0447	1004	MOVB	RCB\$B_ECL_DPX(R0),OBI_B_PRX	: Set default OBI proxy access
004A'CF	03	90	044D	1005	MOVB	#NMA\$C_ACES_BOTH,INT_B_PRX	: Set default internal proxy
			0452	1006			: access state
			0452	1007	:		
			0452	1008	:	Allocate scratch buffer from nonpaged pool for NCB	
			0452	1009	:		
51	007B 8F	3C	0452	1010	MOVZWL	#NET\$C_MAX_NCB+13,R1	: Length of buffer for an NCB
	FBA6'	30	0457	1011	BSBW	NET\$ALONPAGED	: Allocate the buffer
	17 50	E8	045A	1012	BLBS	R0,5\$	: If LBS then block allocated
			045D	1013	:		
			045D	1014	:	Tell NETDRIVER about resource error	
			045D	1015	:		
0008'CF	01	9A	045D	1016	MOVZBL	#NET\$C_DR_RSU,NET_L_REASON	: Reason is 'resource error'
50	0000'CF	D0	0462	1017	MOVL	NET\$GL_PTR_VCB,R0	: Get RCB pointer
			0467	1018	BUMP	W_RCB\$B_CNT_X7E(R0)	: Account for resource error
	34	11	0472	1019	BRB	10\$	: Continue
			0474	1020	:		
			0474	1021	:	Build the NCB and locate the process to accept it	
			0474	1022	:		
0018'CF	52	D0	0474	1023	MOVL	R2,PTR_NCB_BUF	: Save for deallocation
53	0D A2	9E	0479	1024	MOVAB	13(R2),R3	: Get address of string, leave
			047D	1025			: room for count and buf header
0028'CF	53	D0	047D	1026	MOVL	R3,NET_Q_NCB+4	: Store it
0024'CF	53	CE	0482	1027	MNEGL	R3,NET_Q-NCB	: Bias NCB size
	0032	30	0487	1028	BSBW	BUILD_NCB	: Build the NCB
	1B 50	E9	048A	1029	BLBC	R0,10\$	: If LBC then error
0000006E	8F	D1	048D	1030	CMPL	NET_Q_NCB,#NET\$C_MAX_NCB	: Make sure we didn't write
			0496	1031			: past end of buffer
			0496	1032	ASSUME	NET\$C_MAX_NCB LE 255	: Must allow counted string fmt
50	0028'CF	D0	0496	1033	MOVL	NET_Q_NCB+4,R0	: Get ptr to NCB
70	0024'CF	90	049B	1034	MOVB	NET_Q_NCB,-(R0)	: Enter count field and



```
- Process creation
NET$DELIVER_CI - Process and Deliver Inb

0010'CF  50  D0  04A0  1035      MOVL  RO,NET_A_NCB      ; save its address in case NCB
              04A5  1036      ; is to be passed to NETDRIVER
              04A5  1037      ; for a declared name
              0082  30  04A5  1038      BSBW  GET_PROC      ; Find/create process to
              04A8  1039      ; receive the connect
              0440  30  04A8  1040 10$: BSBW  TELL DRV      ; Tell driver about connect
50  0018'CF  D0  04AB  1041      MOVL  PTR_NCB_BUF,RO      ; Address of buffer
      FB4D'  30  0480  1042      BSBW  NET$DEALLOCATE      ; Deallocate the buffer
50  001C'CF  D0  04B3  1043      MOVL  PTR_CON_BUF,RO      ; Address of scratch buffer
      FB45'  30  0488  1044      BSBW  NET$DEALLOCATE      ; Deallocate scratch storage
              05  048B  1045      RSB      ; Done
              04BC  1046
```

```
04BC 1048 SBTTL BUILD_NCB - Build NCB for incoming connect
04BC 1049 +
04BC 1050
04BC 1051 This routine builds the NCB string for the connect, to be later
04BC 1052 given to the destination process (in any number of different ways).
04BC 1053
04BC 1054 Inputs:
04BC 1055
04BC 1056 R6 = XWB address
04BC 1057 NET_Q_NCB = Descriptor of scratch space for NCB
04BC 1058
04BC 1059 Outputs:
04BC 1060
04BC 1061 R0 = status code
04BC 1062 NET_Q_NCB = Descriptor of resultant NCB
04BC 1063 -
04BC 1064 BUILD_NCB: ; Build the NCB
04BC 1065
04BC 1066 Enter 'nodename::'
04BC 1067
53 0028'CF D0 04BC 1068 MOVL NET_Q_NCB+4,R3 ; Get output buffer pointer
58 0000'CF D0 04C1 1069 MOVL NET$GC_CNR_NDI,R11 ; Get root for search
      5A D4 04C6 1070 CLRL R10 ; Indicate no NDI yet
58 3A A6 3C 04C8 1071 MOVZWL XWBSW_REMNOD(R6),R8 ; Get remote node address
      18 50 E9 04CC 1072 $SEARCH egl,ndi,l,tad ; Find NDI with matching address
      0A 50 E9 04DC 1073 BLBC R0,10$ ; If LBC none, use node address
      57 95 04DE 1074 $GETFLD ndi,s,nna ; Get the node name
      06 13 04E7 1075 BLBC R0,10$ ; Invalid if LBC
04EE 1076 TSTB R7 ; Is name null?
04EE 1077 BEQL 10$ ; If EQL use node address
04EE 1078
04EE 1079 Enter ASCII nodename
04EE 1080
63 68 57 28 04EE 1081 MOVCL R7,(R8),(R3) ; Move node name
      07 11 04F2 1082 BRB 20$
04F4 1083
04F4 1084 Enter node address converted to ASCII
04F4 1085
50 3A A6 3C 04F4 1086 10$: MOVZWL XWBSW_REMNOD(R6),R0 ; Get node address
      FB05' 30 04FB 1087 BSBW NET$BIN2ASC ; Move after conversion to ASCII
83 3A3A 8F B0 04FB 1088 20$: MOVW #'A'::',(R3)+ ; Move delimiter
      0500 1089
0500 1090 Enter taskname
0500 1091
50 83 22 90 0500 1092 MOVB #'A'::',(R3)+ ; Enter delimiter
      00BA C6 9A 0503 1093 MOVZBL XWBSW_RPRNAM+1(R6),R0 ; Get object number
      08 13 0508 1094 BEQL 30$ ; If EQL then use taskname
      FAF3' 30 050A 1095 BSBW NET$BIN2ASC ; Else convert to ASCII and move
83 3D 90 050D 1096 MOVB #'A'::',(R3)+ ; Enter delimiter
      19 11 0510 1097 BRB 50$ ; Continue
0512 1098
0512 1099 Enter 0=taskname
0512 1100
83 3D30 8F B0 0512 1101 30$: MOVW #'A'0::',(R3)+ ; Enter 0=
51 00B9 C6 9E 0517 1102 MOVAB XWBSW_RPRNAM(R6),R1 ; Point to process name field
      53 DD 051C 1103 PUSHL R3 ; Save pointer
      0378 30 051E 1104 BSBW GET_PR_NAM ; Move the name text
```

63	68	32	53	8ED0	0521	1105	POPL	R3	:	Recover pointer	
			50	E9	0524	1106	BLBC	R0,60\$	:	If LBC then illegal name format	
			57	28	0527	1107	MOV C3	R7,(R8),(R3)	:	Enter taskname	
					052B	1108	:		:		
					052B	1109	:		:		
					052B	1110	:		:		
					052B	1111	:		:		
83	51	83	2F	90	052B	1111	50\$:	MOVB	#^A'/'',(R3)+	:	Enter delimiter
		000C	CF	80	052E	1112		MOVW	NET_L_LNK,(R3)+	:	Enter local link number
		5B	A6	9E	0533	1113		MOVAB	XWB\$B-DATA(R6),R1	:	Get address of counted data
		50	61	9A	0537	1114		MOVZBL	(R1),R0	:	Get its length
			50	B6	053A	1115		INCW	R0	:	Include its count field
63	11	00	61	50	2C	053C	1116	MOV C5	R0,(R1),#0,#17,(R3)	:	Enter into fixed size field
		51	00A4	C6	9E	0542	1117	MOVAB	XWB\$B_LPRNAM(R6),R1	:	Address local task specifier
			50	81	9A	0547	1118	MOVZBL	(R1)+,R0	:	Get its length
		63	61	50	28	054A	1119	MOV C3	R0,(R1),(R3)	:	Move it
			83	22	90	054E	1120	MOVB	#^A'...',(R3)+	:	Enter terminator
		0024	CF	53	C0	0551	1121	ADDL	R3,NET_Q_NCB	:	Update size in descriptor
			50	01	90	0556	1122	MOVB	#1,R0	:	Indicate success
				05	0559	1123	60\$:	RSB		:	

```
055A 1125 .SBTTL GET_PROC - Locate process to accept connect
055A 1126 +
055A 1127
055A 1128 Find the OBI block associated with the local object. If the OBI is
055A 1129 for a declared name or object then pass the NCB to the declaring
055A 1130 process's mailbox, otherwise create a process to receive the connect.
055A 1131 If there is a server process waiting for more work, then tell the
055A 1132 server process that it can have the connect request.
055A 1133
055A 1134 Inputs:
055A 1135
055A 1136 R6 = XWB address
055A 1137
055A 1138 Own storage
055A 1139
055A 1140 Outputs:
055A 1141
055A 1142 None
055A 1143
055A 1144 GET_PROC:
055A 1145 MOVL NET$GL_CNR_OBI,R11 ; Get process to accept the connect
055A 1146 MOVAB XWB$T_LPRNAM(R6),R1 ; Set up OBI CNR
055A 1147 BSBW GET_PR_ZNA ; Address local task specifier
055A 1148 BLBC RO,TOS ; Get its ZNA field
055A 1149 ; If LBC then format error
055A 1150
055A 1151 Find the OBI CNF
055A 1152
055A 1153
055A 1154 MOVZWL #NET$C_DR_NOBJ,- ; Assume failure due to unknown object
055A 1155 NET_L_REASON
055A 1156 CLRL R10 ; Indicate no current CNF
055A 1157 $SEARCH egl,obi,s,zna ; Find OBI block with this CNF
055A 1158 BLBS RO,20$ ; If LBS then CNF was found
055A 1159 TSTB (R8) ; Is this a numbered object connect ?
055A 1160 BNEQ 10$ ; If NEQ then no such object
055A 1161 MOVQ NET_Q_TASKZNA,R7 ; Else use default TASK ZNA descriptor
055A 1162 MOVZBL S^#NFB$C_OP_EQL,R1 ; Specify match operator
055A 1163 CLRL R10 ; Start from head of list
055A 1164 BSBW CNF$KEY_SEARCH ; Look for the CNF
055A 1165 BLBS RO,25$ ; If LBS then found, br to continue
055A 1166 BRW 100$ ; Complete with error
055A 1167
055A 1168
055A 1169 The OBI CNF has been found. See if the object has been 'declared'
055A 1170 If not, build the .COM file file i.d. and setup its descriptor.
055A 1171
055A 1172
055A 1173 20$:
055A 1174 $GETFLD obi,l,ucb ; Get the associated UCB
055A 1175 BLBC RO,30$ ; If LBC then not declared name
055A 1176 MOVL R8,NET_L_UCB ; Save the UCB pointer
055A 1177 $GETFLD obi,l,pid ; Get the declarer's EPID
055A 1178 BLBC RO,30$ ; If LBC then treat as undeclared
055A 1179 MOVL R8,RO ; Convert from EPID to IPID
055A 1180 JSB G^EXE$EPID_TO_IPID
055A 1181 MOVL RO,NET_L_PID ; Save the PID
055A 1182 MOVZBL #NETUPD$CONNECT,- ; Setup the function code
```

5B 0000'CF D0 055A 1145  
51 00A5 C6 9E 055F 1146  
033C 30 0564 1147  
2B 50 E9 0567 1148  
056A 1149  
056A 1150  
056A 1151  
056A 1152  
056A 1153  
04 3C 056A 1154  
000B'CF 056C 1155  
5A D4 056F 1156  
17 50 E8 0571 1157  
68 95 057E 1158  
10 12 0583 1159  
57 000B'CF 7D 0585 1160  
51 00 9A 058A 1161  
5A D4 058D 1162  
FA6E' 30 058F 1163  
3A 50 E8 0592 1164  
021B 31 0595 1165  
0598 1166  
0598 1167  
0598 1168  
0598 1169  
0598 1170  
0598 1171  
0598 1172  
0598 1173  
52 50 E9 05A3 1174  
0014'CF 58 D0 05A6 1175  
3F 50 E9 05AB 1176  
50 58 D0 05B6 1177  
00000000'GF 16 05B9 1178  
0004'CF 50 D0 05BC 1179  
02 9A 05C2 1180  
05C7 1181



```
0000'CF 01E4 31 05C9 1182
05CC 1183
05CF 1184
05CF 1185
05CF 1186
05CF 1187
05CF 1188
05CF 1189
05CF 1190
05CF 1191
51 00A5 C6 9E 05CF 1192 25$: MOVAB XWBST_LPRNAM(R6),R1 ; Address local task specifier
02C2 30 05D4 1193 BSBW GET_PR_NAM ; Get its name
53 0038'CF D0 05D7 1194 MOVL NET_Q_TSK+4,R3 ; Get address of output buffer
24 68 91 05DC 1195 CMPB (R8),#A'S' ; Does the name start with 'S'?
0C 12 05DF 1196 BNEQ 28$ ; If so,
58 D6 05E1 1197 INCL R8 ; Strip 'S' off front of name
57 D7 05E3 1198 DECL R7
63 0032'CF 28 05E5 1199 MOVC NET_Q_SYSTEM,- ; Prefix name with 'SYS$SYSTEM:'
0036'DF 28 05E9 1200 @NET_Q_SYSTEM+4,(R3)
63 68 57 28 05ED 1201 28$: MOVC3 R7,(R8),(R3) ; Move the name
0034'CF 53 C0 05F1 1202 ADDL R3,NET_Q_TSK ; Update filename size
26 11 05F6 1203 BRB 40$ ; Continue
05F8 1204
05F8 1205
05F8 1206
04 3C 05F8 1207 30$: MOVZWL #NET$C_DR_NOBJ,- ; Assume error
0008'CF 05FA 1208 NET_L_REASON
05FD 1209 $GETFLD obi,s,sti ; Get parsed file id
6B 50 E9 0608 1210 BLBC R0,55$ ; If LBC then file id is invalid
0034'CF 57 7D 060B 1211 MOVQ R7,NET_Q_TSK ; Update filename descriptor
0610 1212
0610 1213
0610 1214
0610 1215
0610 1216
0610 1217
05 50 E8 061B 1218 $GETFLD obi,s,nam ; Get object name for prefix
57 0000'CF 7D 061E 1219 BLBS R0,50$ ; If LBS then name was found
0030'DF 68 57 28 0623 1220 40$: MOVQ NET_Q_NETPREFIX,R7 ; Setup standard prefix descriptor
83 5F 8F 90 0629 1221 50$: MOVQ R7,(R8),@NET_Q_PRC+4 ; Move the prefix
50 000C'CF D0 062D 1222 MOVQ #A' ',(R3)+ ; Move the delimiter
F9CB' 30 0632 1223 MOVL NET_C_LNK,R0 ; Get the local link number
002C'CF 53 C0 0635 1224 BSBW NET$BIN2ASC ; Convert to ascii and append as
063A 1225 the suffix
063A 1226 ADDL R3,NET_Q_PRC ; Done with process name
063A 1227
063A 1228
063A 1229
02 00B9 C6 91 063A 1230 If the connect did not use format type 2, then don't attempt
05 13 063F 1231 a proxy login.
004A'CF 00 90 0641 1232
0646 1233
0646 1234
0646 1235
0646 1236
0646 1237
0646 1238
51$: $GETFLD obi,l,prx ; Get proxy login state
```

```
0049'CF 05 50 E9 0651 1239 BLBC R0,52$ ; If LBC then none specified
58 00CC 58 90 0654 1240 MOVB R8,OBI_B_PRX ; Store it
57 57 9E 0659 1241 52$: MOVAB XWBSB_LOGIN(R6),R8 ; Get address of access info
03 57 9A 065E 1242 MOVZBL (R8)+,R7 ; Get total size
13 13 91 0661 1243 CMPB R7,#3 ; Is it 3 null (counted) strings
00 90 0664 1244 BEQL 60$ ; If so use access info in OBI
004A'CF 00 90 0666 1245 MOVB #NMASC_ACES_NONE,- ; Disallow proxy access
75 8F 57 91 0668 1246 INT_B_PRX ; Store it
13 1B 066B 1247 CMPB R7,#NETSC_MAXACCFD+3 ; Too long?
2B 3C 066F 1248 BLEQU 70$ ; If LEQU then move the strings
0008'CF 3C 0671 1249 MOVZWL #NETSC_DR_IMLONG,- ; Indicate network failure type
013A 31 0673 1250 NET_L_REASON
55$: BRW 100$ ; Continue
60$: $GETFLD obi,s,iac ; Get inbound access control
70$: ;
; Enter the flags word followed by the access control strings
;
53 0040'CF D0 0684 1259 MOVL NET_Q_ACC+4,R3 ; Get pointer to access control buffer
B3 B4 0689 1260 CLRW (R3)+ ; Clear the flags word
068B 1261 $DISPATCH TYPE=B,INT_B_PRX - ; Don't set flag if proxy disallowed
068B 1262 <-
068B 1263 <NMASC_ACES_OUTG, 80$>-
068B 1264 <NMASC_ACES_NONE, 80$>-
068B 1265 >
0697 1266 $DISPATCH TYPE=B,OBI_B_PRX - ; Don't set flag if proxy disallowed
0697 1267 <-
0697 1268 <NMASC_ACES_OUTG, 80$>-
0697 1269 <NMASC_ACES_NONE, 80$>-
0697 1270 >
FE A3 01 A8 06A3 1271 BISW #1,-2(R3) ; Say "proxy login allowed"
63 68 57 28 06A7 1272 80$: MOVCL R7,(R8),(R3) ; Move access control strings,
; even if it's null
003C'CF 53 C0 06AB 1273 ADDL R3,NET_Q_ACC ; Complete string size calc.
024B 30 06B0 1275 BSBW UP_CASE ; Up-case all pertinent strings
;
; Attempt to find an available server process which is waiting
; for a connect which matches it's context.
;
5B 0000'CF D0 06B3 1277 MOVL NET$GL_CNR_SPI,R11 ; Get root of SPI database
5A D4 06B8 1280 CLRL R10 ; Start at beginning of list
58 D4 06BA 1281 81$: CLRL R8 ; Search key is zero
03 50 E8 06BC 1283 $SEARCH neq,spi,l,irp ; Find an SPI with an IRP NE 0
0082 31 06CA 1284 BLBS R0,82$ ; Br if found, check process
34 A6 D5 06CD 1285 BRW 89$ ; Else, create process
14 13 06D0 1286 82$: TSTL XWBSL_PID(R6) ; Is this connect "tagged" for a
; specific process?
D7 50 E9 06D5 1287 BEQL 83$ ; If so, get PID of this server
34 A6 58 D1 06E0 1288 BLBC R0,81$ ; (if not present, error, skip entry)
D1 12 06E3 1290 CMPL R8,XWBSL_PID(R6) ; Is this server the intended process?
83$: BNEQ 81$ ; If not, then continue searching
;
; Always check the access control, even for processes started
; with proxy requested. This way, if different default access
; control is used (each object can specify a unique account,
```

```
06E9 1296      ; including NONE), the wrong process isn't matched.
06E9 1297      ;
06E9 1298      ;
61 50 50 003C 50 E9 06F4 1299      $GETFLD spi,s,acs      ; Get ACS for server process
      68 57 7D 06F7 1300      BLBC RO,81$      ; (if not present, error, skip entry)
      B6 12 2D 06FC 1301      MOVQ NET_Q_ACC,RO      ; Get access string for new connect
      0702 1302      CMPC5 R7,(R8),#0,RO,(R1)      ; Does it match?
      0704 1303      BNEQ 81$      ; If no match, keep searching
      0704 1304      ;
      0704 1305      ; Make sure the process's 'proxy request' flag matches.
      0704 1306      ;
58 0040'DF 01 A8 50 E9 070F 1307      $GETFLD spi,v,prl      ; Get proxy login flag
      00 00 ED 0712 1308      BLBC RO,81$      ; (if not present, error, skip entry)
      9F 12 0719 1309      CMPZV #0,#1,@NET_Q_ACC+4,R8      ; Does proxy login flag match?
      071B 1310      BNEQ 81$      ; If not, try to find another server
      071B 1311      ;
      071B 1312      ; For logical links which request proxy access, require
      071B 1313      ; that the requesting node and username match as well.
      2F 58 E9 071B 1314      BLBC R8,87$      ; If proxy requested,
      8E 50 E9 0729 1315      $GETFLD spi,l,rna      ; Get remote node address for server
58 3A A6 B1 072C 1316      BLBC RO,81$      ; (if not present, error, skip entry)
      88 12 0730 1317      CMPW XWB$W_REMNOD(R6),R8      ; Is it the same node as the connect?
      0F 50 E9 0732 1318      BNEQ 81$      ; If not, try to find another server
      6F A6 9A 073D 1319      $GETFLD spi,s,rld      ; Get remote user ID for server
70 A6 50 00 50 0740 1320      BLBC RO,88$      ; (if not present, error, skip entry)
      68 57 2D 0744 1321      MOVZBL XWB$B_RID(R6),RO      ; Get length of RID for new connect
      02 12 074B 1322      CMPC5 R7,(R8),#0,RO,XWB$T_RID(R6) ; Does it match?
      65 11 074D 1323      BNEQ 88$      ; If no match, then skip it
      FF68 31 074F 1324 87$: BRB SEND_TO_SERVER      ; Server ok, send it the connect
      0752 1325      BRW 81$      ; (Branch helper to top of loop)
      0752 1326 88$:
      0752 1327 89$:
      0752 1328
      0752 1329
      0752 1330
      0752 1331
      0752 1332
      0752 1333
      0752 1334
      0752 1335
      0752 1336
      0752 1337
      0752 1338
      0752 1339
      0752 1340
      0752 1341
      078E 1342
      07 50 E8 078E 1343      BLBS RO,90$
      01 3C 0791 1344      MOVZWL #NET$C_DR_RSU,-
      0008'CF 11 0793 1345      NET_L_REASON
      1B 11 0796 1346      BRB 100$
50 0004'CF D0 0798 1347 90$: MOVL NET_L_PID,RO
      58 50 D0 079D 1348      MOVL RO,R8
      00000000'GF 16 07A0 1349      JSB G^EXESEPID TO_I^PID
      0004'CF 50 D0 07A6 1350      MOVL RO,NET_L_PID
      04 3C 07AB 1351      MOVZWL #NETUPD$-PROCRE,-
      0000'CF 07AD 1352      NET_L_FCT
      ;
      ; create a process
      ; Network NETSERVER.COM filename
      ; Access control strings
      ; 1st NCB (solely for LOGIN proxy use,
      ; Process name
      ; Image (LOGINOUT) to run first
      ; Place to store process id
      ; Priority
      ; UIC is [10,40]
      ; This is a network process
      ; MBX for termination
      ; notification
      ; If LBS process was created
      ; Assume because couldn't get
      ; the resources
      ; Take common exit
      ; Get the EPID returned by CREPRC
      ; Save EPID
      ; Convert to internal PID format
      ; Use internal format of PID
      ; Say 'process created'
```



```

0081  30  0780  1353  :
      0780  1354  : The network process is created. Now create an SPI database entry
      0780  1355  : so we can keep track of it.
      0780  1356  :
      0780  1357  BSBW  CREATE_SPI      ; Create SPI database entry
      0783  1358  : Ignore errors if can't be inserted
05  0783  1359  100$: RSB              ; Common exit

```

```
07B4 1361 .SBTTL SEND_TO_SERVER - Send connect to waiting server
07B4 1362 :
07B4 1363 :
07B4 1364 : There is a waiting server which can handle the incoming connect. Set
07B4 1365 : it up so that the server can accept the logical link.
07B4 1366 :
07B4 1367 : Inputs:
07B4 1368 :
07B4 1369 :     R11 = SPI CNR address
07B4 1370 :     R10 = CNF for server database entry
07B4 1371 :
07B4 1372 :
07B4 1373 SEND_TO_SERVER:
56 5A DO 07B4 1374 MOVL R10,R6 ; Save address of old CNF
F846' 30 07B7 1375 BSBW NET$GETUTLBUF ; Get permission to use utility buffer
F843' 30 07BA 1376 BSBW CNF$INIT_UTL ; Initialize utility buffer
58 56 DO 07BD 1377 MOVL R6,R8 ; Pass address of old CNF
F83D' 30 07C0 1378 BSBW CNF$COPY ; Copy old CNF to new CNF space
57 0024'CF 7D 07C3 1379 MOVQ NET_Q_NCB,R7 ; Get descriptor of NCB
07C8 1380 $PUTFLD spi,s,ncb ; Store it
57 0034'CF 7D 07D3 1381 MOVQ NET_Q_TSK,R7 ; Get procedure filespec
07D8 1382 $PUTFLD spi,s,sfi ; Store it
57 002C'CF 7D 07E3 1383 MOVQ NET_Q_PRC,R7 ; Get process name
07E8 1384 $PUTFLD spi,s,pnm ; Store it
F80A' 30 07F3 1385 BSBW CNF$INSERT ; Insert new CNF (R10 = UTILBUF)
07F6 1386 ; and delete old CNF (R6)
07F6 1387 ; returns: R10 = valid CNF
07F6 1388 $GETFLD spi,l,pid ; Get PID of server process
0004'CF 58 DO 0801 1389 MOVL R8,NET_L_PID ; Make it seem as if it was just created
0806 1390 $GETFLD spi,l,irp ; Get waiting DECLSERV IRP
F7EC' 30 0811 1391 BSBW CNF$CLR_FIELD ; and clear it from database
53 58 DO 0814 1392 MOVL R8,R3 ; Get IRP address
38 A3 00' DO 0817 1393 MOVL S^SS$NORMAL,IRP$L_IOST1(R3) ; Set success into IRP
3C A3 0004'CF DO 081B 1394 MOVL NET_L_PID,IRP$L_IOST2(R3) ; Return IPID of SPI process as well
55 1C A3 DO 0821 1395 MOVL IRP$L_UCB(R3),R5 ; Get UCB address
00000000'GF 16 0825 1396 JSB G^COM$POST ; and complete the request
F7D2' 30 082B 1397 BSBW NET$DEC TRANS ; Account for completed transaction
0000'CF 04 3C 082E 1398 MOVZWL #NETUPD$_PROC,NET_L_FCT ; Tell NETDRIVER that process exists
05 0833 1399 RSB
```

```
0834 1401 .SBTTL CREATE_SPI - Create SPI database entry
0834 1402 +
0834 1403
0834 1404 Subroutine to create an SPI database entry after having just created
0834 1405 the network process.
0834 1406
0834 1407 Inputs:
0834 1408
0834 1409 R6 = XWB address
0834 1410 Own storage
0834 1411
0834 1412 Outputs:
0834 1413
0834 1414 R0 = Status code
0834 1415 -
0834 1416 CREATE_SPI:
0834 1417 BSBW NET$GETUTLBUF ; Get permission to use utility buffer
58 0000'CF 30 0837 1418 MOVL NET$GL_CNR_SPI,R11 ; Get root of SPI database
0834 1419 BSBW CNF$INIT_UTL ; Init utility buffer as a CNF block
58 0004'CF D0 083F 1420 MOVL NET_L_PID,R8 ; Get PID of server process
0834 1421 $PUTFLD spi,l,pid ; Store parameter into entry
58 0040'DF 01 00 EF 084F 1422 EXTZV #0,#1,NET_Q_ACC+4,R8 ; Get proxy flag sent to LOGIN
0834 1423 $PUTFLD spi,v,prl ; Store it
57 003C'CF 7D 0861 1424 MOVQ NET_Q_ACC,R7 ; Get access control sent to LOGIN
0834 1425 $PUTFLD spi,s,acs ; Store ACS string sent to LOGIN
58 3A A6 3C 0871 1426 MOVZWL XWB$W_REMNOD(R6),R8 ; Get remote node address
0834 1427 $PUTFLD spi,l,rna ; Store it
57 6F A6 9A 0880 1428 MOVZBL XWB$B_RID(R6),R7 ; Make descriptor of RID
58 70 A6 9E C884 1429 MOVAB XWB$T_RID(R6),R8
0834 1430 $PUTFLD spi,s,rid ; Store it
0834 1431 CLRL R6 ; No 'old' CNF entry
0834 1432 BSBW CNF$INSERT ; Insert into database
0834 1433 RSB
```



```
0899 1435 .SBTTL GET_PR_NAM      - Get name of object procedure
0899 1436 .SBTTL GET_PR_ZNA     - Construct ZNA string for an object
0899 1437 :+
0899 1438 :
0899 1439 : Inputs:
0899 1440 :
0899 1441 :     R1 = Address of local task specifier
0899 1442 :
0899 1443 : Outputs:
0899 1444 :
0899 1445 :     R7/R8 = Descriptor of resultant string
0899 1446 :-
0899 1447 :
0899 1448 :
0899 1449 GET_PR_NAM: .ENABL LSB
0899 1450 MOVAB ZNABUF,R8          ; Get procedure name
0899 1451 MOVL R8,R3              ; Setup buffer pointer
0899 1452 BRB 5$                ; Make a copy
0899 1453 GET_PR_ZNA:          ; Continue
0899 1454 MOVAB ZNABUF,R8          ; Point to ZNA buffer
0899 1455 MOVL R8,R3              ; Make a copy
0899 1456 MOVB 1(R1),(R3)+        ; Enter object type
0899 1457 5$: CVTWB (R1)+,R0      ; Get format type, skip over object type
0899 1458 BNEQ 20$                ; If NEQ then not numbered object
0899 1459 10$: TSTB -1(R1)        ; Is object type zero ?
0899 1460 BEQL 40$                ; If EQL then error
0899 1461 BRB 60$                ; Else we're done
0899 1462 20$: TSTB -1(R1)        ; Is object type zero ?
0899 1463 BNEQ 40$                ; If NEQ then error
0899 1464 CMPB #1,R0              ; Format type 1 is a counted string
0899 1465 BEQL 30$                ; If EQL then go move the string
0899 1466 CMPB #2,R0              ; Format type 2 is UIC + counted string
0899 1467 BNEQ 40$                ; If NEQ then format type is unknown
0899 1468 TSTL (R1)+              ; Skip over UIC
0899 1469 30$: MOVZBL (R1)+,R0    ; Get taskname string size
0899 1470 BEQL 40$                ; If EQL then illegal format
0899 1471 CMPB R0,#MAX_TASKNAM    ; Is it within bounds?
0899 1472 BLEQU 50$              ; If LEQU then legal format
0899 1473 40$: CLRL R0           ; Else, indicate error
0899 1474 MOVZWL #NET$C_DR_FMT,-  ; Setup network failure code
0899 1475 NET_L_REASON
0899 1476 BRB 70$                ; Take common exit
0899 1477 50$: MOVCL R0,(R1),(R3) ; Enter string
0899 1478 60$: SUBL3 R8,R3,R7    ; Get string size
0899 1479 MOVL #1,R0            ; Indicate success
0899 1480 70$: RSB
0899 1481
0899 1482 .DSABL LSB
```

```
08EB 1484 .SBTTL TELL_DRV          - Call NETDRIVER
08EB 1485 :++
08EB 1486 :
08EB 1487 : Call NETDRIVER to perform a given function.
08EB 1488 :
08EB 1489 : Inputs:
08EB 1490 :
08EB 1491 : NET_L_R0-R5 = Arguments to NETDRIVER function
08EB 1492 :--
08EB 1493 TELL_DRV:
50 0000'CF 7D 08EB 1494 MOVQ NET_L_R0,R0          ; Tell driver about process
52 0008'CF 7D 08F0 1495 MOVQ NET_L_R2,R2          ; Get regs for call
54 0010'CF 7D 08F5 1496 MOVQ NET_L_R4,R4          ;
    F703' 30 08FA 1497 BSBW CALL_NETDRIVER        ; Call driver
    05 08FD 1498 RSB
```

```
08FE 1500 .SBTTL UP_CASE - Upcase the LOGINOUT strings
08FE 1501 +
08FE 1502 :
08FE 1503 : The NCB (up to the '/'), the access control strings, the taskname, and the
08FE 1504 : process name are up-cased in place.
08FE 1505 :
08FE 1506 : INPUTS: none
08FE 1507 :
08FE 1508 : OUTPUTS: none
08FE 1509 :
08FE 1510 : All register contents are preserved.
08FE 1511 :
08FE 1512 :
08FE 1513 : UP_CASE:
55 0000 3F BB 08FE 1514 : PUSH R0,R1,R2,R3,R4,R5 ; Up-case strings passed to LOGINOUT
54 2F 9E 0900 1515 : MOVAB NET$AB_UPASCNUM,R5 ; Save regs
53 0024 CF 9E 0905 1516 : MOVAB #A'/' ,R4 ; Get translation table
3C 10 0908 1517 : MOVAB NET_Q_NCB,R3 ; Setup terminator
54 D4 090D 1518 : BSBB UP_IT ; Point to NCB descriptor
53 0034 CF 9E 0911 1519 : CLRL R4 ; Up-case it in place
33 10 0916 1520 : MOVAB NET_Q_TSK,R5 ; Say "no terminator"
53 002C CF 9E 0918 1521 : BSBB UP_IT ; Point to task-name descriptor
2C 10 091D 1522 : MOVAB NET_Q_PRC,R3 ; Up-case it in place
53 003C CF 9E 091F 1523 : BSBB UP_IT ; Point to process-name descriptor
51 04 A3 D0 0924 1524 : MOVAB NET_Q_ACC,R3 ; Up-case it in place
51 02 C0 0928 1525 : MOVL 4(R3),R1 ; Get access control descriptor
52 81 9A 092B 1526 : ADDL #2,R1 ; Get pointer to strings
36 10 092E 1527 : MOVZBL (R1)+,R2 ; Skip over flags word
52 81 9A 0930 1528 : BSBB UP_CASE_LOOP ; Get count of bytes in username
31 10 092E 1529 : MOVZBL (RT)+,R2 ; Start at end of loop
52 81 9A 0933 1530 : BSBB UP_CASE_LOOP ; Get count of bytes in password
2C 10 0933 1531 : MOVZBL (RT)+,R2 ; Start at end of loop
7E 04 A3 63 C1 0938 1532 : BSBB UP_CASE_LOOP ; Get count of bytes in account name
8E 51 D1 093A 1533 : ADDL3 (R3),4(R3),-(SP) ; Start at end of loop
03 1A 093F 1534 : CMPL R1,(SP)+ ; Get address of end of strings
3F BA 0942 1535 : BGTRU 10$ ; Have we gone too far?
05 05 0944 1536 : POPR #M<R0,R1,R2,R3,R4,R5> ; If GTRU then yes
0946 1537 : RSB ; Restore regs
0947 1538 :
0947 1539 :
0947 1540 10$: BUG_CHECK NETNOSTATE,FATAL ; Access control strings setup
0948 1541 : ; incorrectly
0948 1542 :
0948 1543 :
0948 1544 : .ENABL LSB
0948 1545 :
51 52 63 3C 0948 1546 UP_IT: MOVZWL (R3),R2 ; Get string length
04 A3 D0 094E 1547 : MOVL 4(R3),R1 ; Point to string
12 11 0952 1548 : BRB UP_CASE_LOOP ; Start at end of loop
50 81 90 0954 1549 20$: MOVAB (RT)+,R0 ; Get next character
54 50 91 0957 1550 : CMPB R0,R4 ; Is it the terminator?
0D 13 095A 1551 : BEQL 60$ ; If EQL yes, we're done
50 6540 90 095C 1552 : MOVAB (R5)[R0],R0 ; Up-case it
04 13 0960 1553 : BEQL UP_CASE_LOOP ; If EQL then not alpha-numeric
FF A1 50 90 0962 1554 : MOVAB R0,-1(RT) ; Store up-cased value
0966 1555 :
0966 1556 UP_CASE_LOOP:
```



NETPROCRE  
V04-000

- Process creation  
UP\_CASE - Upcase the LOGINOUT strings

J 16

16-SEP-1984 01:27:29  
5-SEP-1984 02:21:33

VAX/VMS Macro V04-00  
[NETACP.SRC]NETPROCRE.MAR;1

Page 37  
(20)

```
EB 52  F4 0966 1557      SOBGEQ R2,20$      ; Loop for each character
        05 0969 1558 60$: RSB                ; Done
        096A 1559
        096A 1560      .DSABL LSB
        096A 1561
        096A 1562
        096A 1563 .END
```

NETPROC  
Symbol table

## - Process creation

K 16

16-SEP-1984 01:27:29 VAX/VMS Macro V04-00  
5-SEP-1984 02:21:33 [NETACP.SRC]NETPROC.MAR;1Page 38  
(20)

```

$ST1      = 00000000
ACC       = 0000000C
ACCSK_TERMLEN = 00000054
ACPSC_STA_F = 00000004
ACPSC_STA_H = 00000005
ACPSC_STA_I = 00000000
ACPSC_STA_N = 00000001
ACPSC_STA_R = 00000002
ACPSC_STA_S = 00000003
BBUF      = 0000010E R      02
BIT...    = 00000006
BUGS_ACPMBFAIL = ***** X      04
BUGS_NETNOSTATE = ***** X      05
BUILD_NCB = 0000048C R      04
CALL_NETDRIVER = ***** X      04
CNF       = 00000024
CNFSCLR_FIELD = ***** X      04
CNFSCOPY  = ***** X      04
CNFSC_LENGTH = 00000024
CNFSDELETE = ***** X      05
CNFSGET_FIELD = ***** X      04
CNFSINIT_UTL = ***** X      04
CNFSINSERT = ***** X      04
CNFSKEY_SEARCH = ***** X      05
CNFSPURGE = ***** X      05
CNFSPUT_FIELD = ***** X      05
CNFS_ADVANCE = 00000000
CNFS_QUIT   = 00000002
CNFS_TAKE_CURR = 00000003
CNFS_TAKE_PREV = 00000001
COMSPOST   = ***** X      04
CONNECT    = 000001D7 R      04
CONN_SPACE = 000003E8
CREATE_LLI = 00000000 R      04
CREATE_SPI = 00000834 R      04
DEAL_XQB   = 00000090 R      05
DELPROC    = 000001E2 R      04
DET_AB_ACC = 00000044 R      02
DET_C_ACC  = 00000005
DLESPRC_EXIT = ***** X      04
ENDBUF     = 0000011C R      02
EXESEPID_TO_IPID = ***** X      04
EXIT_BUF   = 00000018 R      03
EXIT_CODE  = 0000005C R      02
EXIT_ID    = 00000058 R      02
EXIT_MSG   = 00000058 R      02
GET_PROC   = 0000055A R      04
GET_PR_NAM = 00000899 R      04
GET_PR_ZNA = 000008A3 R      04
INT_B_PRX  = 0000004A R      02
IOS_READVBLK = ***** X      04
IRPSL_IOST1 = 00000038
IRPSL_IOST2 = 0000003C
IRPSL_UCB  = 0000001C
LGIS_INVPWD = ***** X      04
LLISZ_NDC_LZ = 00000024
LLISZ_NDC_RT = 00000008
```

```

LSB       = 00000000
LSBSB_R_CXBCNT = 00000028
LSBSB_R_CXBQUO = 00000029
LSBSB_SPARE = 0000002A
LSBSB_STS  = 0000002B
LSBSB_X_ADJ = 0000000B
LSBSB_X_CXBACT = 0000000D
LSBSB_X_CXBCNT = 0000000F
LSBSB_X_CXBQUO = 0000000E
LSBSB_X_PKTWND = 0000000C
LSBSB_X_REQ = 0000000A
LSBSL_CROSS = 0000002C
LSBSL_R_CXB = 00000020
LSBSL_R_IRP = 0000001C
LSBSL_X_CXB = 00000018
LSBSL_X_IRP = 00000014
LSBSL_X_PND = 00000010
LSBSM_BOM  = 00000020
LSBSM_EOM  = 00000040
LSBSM_LI   = 00000001
LSBSS_LSB  = 00000030
LSBSS_SPARE = 00000004
LSBSS_STS  = 00000001
LSBSV_BOM  = 00000005
LSBSV_EOM  = 00000006
LSBSV_LI   = 00000000
LSBSV_SPARE = 00000001
LSBSW_HAA  = 00000008
LSBSW_HAR  = 00000006
LSBSW_HAX  = 00000026
LSBSW_HNR  = 00000024
LSBSW_HXS  = 00000004
LSBSW_LNX  = 00000002
LSBSW_LUX  = 00000000
LTBSL_SLOTS = 00000010
LTBSL_SLT_NXT = 00000000
LTBSL_XWB  = 0000000C
MAX_TASKNAM = 0000000C
MBX_ACTION = 000000E6 R      04
MBX_CHAN   = 0000004C R R      02
MBX_IOSB   = 00000050 R R      02
MBX_LEN    = 00000052 R      02
MBX_MSG_LTH = 00000096
MBX_PID    = 00000054 R      02
MBX_RDCNT  = 0000004E R R      02
MBX_UNIT   = 0000011A R      02
MSG$_CONNECT = 00000032
MSG$_DELPROC = 00000003
MSG$_PATHLOST = 00000036
MSG$_RESET  = 00000041
NCB_DATA   = 0000005C R      02
NDCSC_LENGTH = 0000001C
NET$AB_UPASCNUM = ***** X      04
NET$ACQUIRE_NDCOU = ***** X      04
NET$ALLOCATE = ***** X      04
NET$ALONPAGED = ***** X      04
NET$BIN2ASC = ***** X      04
```

NETPROCRE  
Symbol table

- Process creation

L 16

16-SEP-1984 01:27:29 VAX/VMS Macro V04-00  
5-SEP-1984 02:21:33 [NETACP.SRC]NETPROCRE.MAR;1

Page 39  
(20)

NET\$CONNECT_FAIL	0000024C	RG	04
NET\$CREATE_MBX	0000004C	RG	04
NET\$C_ACT_TIMER	= 0000001E		
NET\$C_DR_ACCESS	= 00000022		
NET\$C_DR_EXIT	= 00000026		
NET\$C_DR_FMT	= 00000005		
NET\$C_DR_IMLONG	= 0000002B		
NET\$C_DR_NOBJ	= 00000004		
NET\$C_DR_RSU	= 00000001		
NET\$C_DYN_WQE	*****	X	02
NET\$C_EFN_ASYN	= 00000002		
NET\$C_EFN_WAIT	= 00000001		
NET\$C_IPL	= 00000008		
NET\$C_MAXACFLD	= 00000027		
NET\$C_MAXLINNAM	= 0000000F		
NET\$C_MAXLNK	= 000003FF		
NET\$C_MAXNODNAM	= 00000006		
NET\$C_MAXOBJNAM	= 0000000C		
NET\$C_MAX_AREAS	= 0000003F		
NET\$C_MAX_LINES	= 00000040		
NET\$C_MAX_NCB	= 0000006E		
NET\$C_MAX_NODES	= 000003FF		
NET\$C_MAX_OBJ	= 000000FF		
NET\$C_MAX_WQE	= 00000014		
NET\$C_MINBUFSIZ	= 000000C0		
NET\$C_TID_ACT	= 00000003		
NET\$C_TID_RUS	= 00000001		
NET\$C_TID_XRT	= 00000002		
NET\$C_TRCTL_CEL	= 00000002		
NET\$C_TRCTL_OVR	= 00000005		
NET\$C_UTLBUFSIZ	= 00001000		
NET\$DEALLOCATE	*****	X	05
NET\$DECR_MCOUNT	*****	X	05
NET\$DEC_TRANS	*****	X	04
NET\$DELIVER_CI	000003DC	R	04
NET\$DLL_X25_CALL	*****	X	04
NET\$DLL_X25_RESET	*****	X	04
NET\$DRV_CANCEL	*****	X	04
NET\$FLUSH_LLI_CNT	*****	X	05
NET\$GETUTCBUF	*****	X	04
NET\$GL_CNR_LLI	*****	X	05
NET\$GL_CNR_NDI	*****	X	04
NET\$GL_CNR_OBI	*****	X	04
NET\$GL_CNR_SPI	*****	X	04
NET\$GL_NET_UCB	*****	X	04
NET\$GL_PTR_VCB	*****	X	04
NET\$GO_MBX_NAME	00000020	RG	03
NET\$GO_WQE_MBX	000000F6	RG	02
NET\$KICL_MBX	0000008B	RG	04
NET\$MBX_AST	000000EE	RG	04
NET\$MBX_QIO	00000098	RG	04
NET\$M_MAXLNKMSK	= 000003FF		
NET\$PROC_XWB	00000000	RG	05
NET\$RELEASE_NDCOU	*****	X	05
NET\$RESEND_SERVER	000002CA	RG	04
NET\$SCAN_FOR_ZNA	00000287	RG	04
NET\$SERVER_FAIL	0000026D	RG	04

NET\$SET_MBX_AST	000000C9	RG	04
NET\$STARTUP_OBJ	00000317	RG	04
NET\$STARTUP_OBJ_NAM	000002FD	RG	04
NETUPDS_ABORT	= 00000001		
NETUPDS_CONNECT	= 00000002		
NETUPDS_EXIT	= 00000003		
NETUPDS_PROCRE	= 00000004		
NET_A_LCI	00000020	R	02
NET_A_NCB	00000010	R	02
NET_L_FCT	00000000	R	02
NET_L_LNK	0000000C	R	02
NET_L_LPD	00000004	R	02
NET_L_PID	00000004	R	02
NET_L_R0	00000000	R	02
NET_L_R1	00000004	R	02
NET_L_R2	00000008	R	02
NET_L_R3	0000000C	R	02
NET_L_R4	00000010	R	02
NET_L_R5	00000014	R	02
NET_L_REASON	00000008	R	02
NET_L_UCB	00000014	R	02
NET_Q_ACC	0000003C	R	02
NET_Q_IMAGE	00000045	R	03
NET_Q_NCB	00000024	R	02
NET_Q_NETPREFIX	00000000	R	03
NET_Q_PRC	0000002C	R	02
NET_Q_PROG	0000005B	R	03
NET_Q_SYSTEM	00000032	R	03
NET_Q_TASKZNA	0000000B	R	03
NET_Q_TSK	00000034	R	02
NEW_LNK	0000009E	R	05
NFBSC_LLI_XWB	= 08010017		
NFBSC_NDI_NNA	= 02020043		
NFBSC_NDI_TAD	= 02010010		
NFBSC_OBI_IAC	= 03020043		
NFBSC_OBI_NAM	= 03020044		
NFBSC_OBI_NUM	= 03010014		
NFBSC_OBI_PID	= 03010015		
NFBSC_OBI_PRX	= 03010016		
NFBSC_OBI_SFI	= 03020042		
NFBSC_OBI_UCB	= 03010012		
NFBSC_OBI_ZNA	= 03020041		
NFBSC_OP_EQL	= 00000000		
NFBSC_OP_NEQ	= 00000003		
NFBSC_SPI_ACS	= 12020041		
NFBSC_SPI_IRP	= 12010011		
NFBSC_SPI_NCB	= 12020044		
NFBSC_SPI_PID	= 12010010		
NFBSC_SPI_PNM	= 12020045		
NFBSC_SPI_PRL	= 12000002		
NFBSC_SPI_RID	= 12020042		
NFBSC_SPI_RNA	= 12010013		
NFBSC_SPI_SFI	= 12020043		
NMASC_ACES_BOTH	= 00000003		
NMASC_ACES_NONE	= 00000000		
NMASC_ACES_OUTG	= 00000002		
NSPSC_EXT_CNK	= 0000001E		

NETPROC  
Symbol table

- Process creation

M 16

16-SEP-1984 01:27:29 VAX/VMS Macro V04-00  
5-SEP-1984 02:21:33 [NETACP.SRC]NETPROC.MAR;1

Page 40  
(20)

```

NSP$C_MAXHDR      = 00000009
OBI_B-PRX         = 00000049 R    02
PRS-IPL           ***** X    05
PRC               = 000000C8
PTR_CON_BUF       = 0000001C R    02
PTR_NCB_BUF       = 00000018 R    02
RCB$B_ECL_DPX     = 00000067
RCB$B_PTR_LTB     = 00000024
RCB$B_CNT_XRE     = 0000009C
RMS$FNF           ***** X    04
SEND_TO_SERVER    = 000007B4 R    04
SIZ...           = 00000001
SS$ABORT          ***** X    04
SS$CANCEL         ***** X    04
SS$CONNECFAIL     ***** X    05
SS$NORMAL         ***** X    04
ST$V_INHIB_MSG    = 0000001C
ST$M_NETLOG       = 00000080
ST$M_NOACNT       = 00000008
ST$M_NOAUTH       = 00000040
SYSSCREMBX        ***** GX   04
SYSSCREPRC        ***** GX   04
SYSSDASSGN        ***** GX   04
SYSSGB_DEFPR1     ***** X    04
SYSSGETCHN        ***** GX   04
SYSSQIO           ***** GX   04
TASKZNA           = 00000013 R    03
TELL_DRV          = 000008EB R    04
TRSC_MAXHDR       = 0000001C
TRSC_NI_ALLEND1   = 040000AB
TRSC_NI_ALLEND2   = 00000000
TRSC_NI_ALLROU1   = 030000AB
TRSC_NI_ALLROU2   = 00000000
TRSC_NI_PREFIX    = 000400AA
TRSC_NI_PROT      = 00000360
TRSC_PRT_ECL      = 0000001F
TRSC_PRI_RTHRU    = 0000001F
TSK               = 0000012C
UP_CASE           = 000008FE R    04
UP_CASE_LOOP      = 00000966 R    04
UP_IT             = 0000094B R    04
WQESC_SUB_MBX     = 00000005
WQESINSQUE        ***** X    04
WQESL_PM1         = 00000010
WQESL_PH2         = 00000014
WQE_MBX_LTH       = 00000018
X25_DEV_NAME      = 00000077 R    03
XWB               = 00000000
XWB$B_ACCESS      = 0000000B
XWB$B_DATA        = 0000005B
XWB$B_FIPL        = 0000001F
XWB$B_LOGIN       = 000000CC
XWB$B_LPRNAM      = 000000A4
XWB$B_PRO         = 0000005A
XWB$B_RID         = 0000006F
XWB$B_RPRNAM      = 000000B8
XWB$B_SP3         = 0000006E

```

```

XWB$B_STA         = 0000001E
XWB$B_TYPE        = 0000000A
XWB$B_X_FLW       = 0000006C
XWB$B_X_FLWCNT    = 0000006D
XWB$C_COMLNG      = 000000A4
XWB$C_CONLNG      = 00000112
XWB$C_DATA        = 00000010
XWB$C_LOGIN       = 00000040
XWB$C_LPRNAM      = 00000014
XWB$C_NDC_LNG     = 00000020
XWB$C_NUMSTA      = 00000008
XWB$C_RID         = 00000010
XWB$C_RPRNAM      = 00000014
XWB$C_STA_CAR     = 00000002
XWB$C_STA_CCS     = 00000004
XWB$C_STA_CIR     = 00000003
XWB$C_STA_CIS     = 00000001
XWB$C_STA_CLO     = 00000000
XWB$C_STA_DIR     = 00000006
XWB$C_STA_DIS     = 00000007
XWB$C_STA_RUN     = 00000005
XWB$C_DEA_IRP     = 00000104
XWB$C_FPC         = 00000020
XWB$C_FR3         = 00000024
XWB$C_FR4         = 00000028
XWB$C_ICB         = 0000010C
XWB$C_IRP_ACC     = 00000080
XWB$C_LINK        = 0000002C
XWB$C_ORGUCB      = 00000010
XWB$C_PID         = 00000034
XWB$C_VCB         = 00000030
XWB$C_WLBL        = 00000004
XWB$C_WLFL        = 00000000
XWB$M_FLG_BREAK   = 00000001
XWB$M_FLG_CLO     = 00000200
XWB$M_FLG_IAVL    = 00001000
XWB$M_FLG_SCD     = 00000100
XWB$M_FLG_SDACK   = 00000008
XWB$M_FLG_SDFL    = 00004000
XWB$M_FLG_SDT     = 00000080
XWB$M_FLG_SIACK   = 00000004
XWB$M_FLG_SIFL    = 00002000
XWB$M_FLG_SLI     = 00000010
XWB$M_FLG_TBPR    = 00000800
XWB$M_FLG_WBP     = 00000040
XWB$M_FLG_WBUF    = 00000002
XWB$M_FLG_WDAT    = 00000400
XWB$M_FLG_WHGL    = 00000020
XWB$M_PRO_CCA     = 00000008
XWB$M_PRO_NAR     = 00000010
XWB$M_PRO_NFC     = 00000001
XWB$M_PRO_PH2     = 00000004
XWB$M_PRO_SFC     = 00000002
XWB$M_STS_ASTPND   = 00000400
XWB$M_STS_ASTREQ   = 00000800
XWB$M_STS_CON      = 00000010
XWB$M_STS_DIS     = 00000008

```



NETPROC  
Symbol table

- Process creation

B 1

16-SEP-1984 01:27:29 VAX/VMS Macro V04-00  
5-SEP-1984 02:21:33 [NETACP.SRC]NETPROCRE.MAR;1Page 41  
(20)

XWBSM\_STS\_DTNK = 00000100  
XWBSM\_STS\_LINAK = 00000200  
XWBSM\_STS\_NDC = 00001000  
XWBSM\_STS\_OVF = 00000080  
XWBSM\_STS\_RBP = 00000040  
XWBSM\_STS\_SOL = 00000004  
XWBSM\_STS\_TID = 00000001  
XWBSM\_STS\_TLI = 00000002  
XWBSM\_STS\_TMO = 00000020  
XWBSQ\_FORK = 00000014  
XWBSQ\_FREE\_CXB = 00000118  
XWBSR\_CON\_BLK = 000000A4  
XWBSR\_RUN\_BLK = 000000A4  
XWBS = 00000006  
XWBS\_COMLNG = 0000006E  
XWBS\_CON\_BLK = 0000006E  
XWBS\_DATA = 00000010  
XWBS\_DT = 00000030  
XWBS\_FLG = 00000002  
XWBS\_FORK = 00000008  
XWBS\_FREE\_CXB = 00000008  
XWBS\_LI = 00000030  
XWBS\_LOGIN = 0000003F  
XWBS\_LPRNAM = 00000013  
XWBS\_NDC = 00000020  
XWBS\_PRO = 00000001  
XWBS\_RID = 00000010  
XWBS\_RPRNAM = 00000013  
XWBS\_RUN\_BLK = 00000064  
XWBS\_STS = 00000002  
XWBS\_XWB = 00000120  
XWBS = 00000112  
XWBS\_DATA = 0000005C  
XWBS\_DT = 000000A4  
XWBS\_LI = 000000D4  
XWBS\_LOGIN = 000000CD  
XWBS\_LPRNAM = 000000A5  
XWBS\_RID = 00000070  
XWBS\_RPRNAM = 000000B9  
XWBSV\_FLG\_BREAK = 00000000  
XWBSV\_FLG\_CLO = 00000009  
XWBSV\_FLG\_I AVL = 0000000C  
XWBSV\_FLG\_SCD = 00000008  
XWBSV\_FLG\_SDACK = 00000003  
XWBSV\_FLG\_SDFL = 0000000E  
XWBSV\_FLG\_SDT = 00000007  
XWBSV\_FLG\_SIAK = 00000002  
XWBSV\_FLG\_SIFL = 0000000D  
XWBSV\_FLG\_SLI = 00000004  
XWBSV\_FLG\_TBPR = 0000000B  
XWBSV\_FLG\_WBP = 00000006  
XWBSV\_FLG\_WBUF = 00000001  
XWBSV\_FLG\_WDAT = 0000000A  
XWBSV\_FLG\_WHGL = 00000005  
XWBSV\_PRO\_CCA = 00000003  
XWBSV\_PRO\_NAR = 00000004  
XWBSV\_PRO\_NFC = 00000000

XWBSV\_PRO\_PH2 = 00000002  
XWBSV\_PRO\_SFC = 00000001  
XWBSV\_STS\_ASTPND = 0000000A  
XWBSV\_STS\_ASTREQ = 0000000B  
XWBSV\_STS\_CON = 00000004  
XWBSV\_STS\_DIS = 00000003  
XWBSV\_STS\_DTNK = 00000008  
XWBSV\_STS\_LINAK = 00000009  
XWBSV\_STS\_NDC = 0000000C  
XWBSV\_STS\_OVF = 00000007  
XWBSV\_STS\_RBP = 00000006  
XWBSV\_STS\_SOL = 00000002  
XWBSV\_STS\_TID = 00000000  
XWBSV\_STS\_TLI = 00000001  
XWBSV\_STS\_TMO = 00000005  
XWBSW\_CI\_PATH = 00000110  
XWBSW\_DELAY = 0000004E  
XWBSW\_DLY\_FACT = 00000056  
XWBSW\_DLY\_WGHT = 00000058  
XWBSW\_ELAPSE = 0000004A  
XWBSW\_FLG = 0000001C  
XWBSW\_LOCLNK = 0000003E  
XWBSW\_LOCSIZ = 00000040  
XWBSW\_PATH = 00000038  
XWBSW\_PROGRESS = 00000052  
XWBSW\_REFCNT = 0000000C  
XWBSW\_REMLNK = 0000003C  
XWBSW\_REMNOD = 0000003A  
XWBSW\_REMSIZ = 00000042  
XWBSW\_RETRAN = 00000054  
XWBSW\_R\_REASON = 00000044  
XWBSW\_SIZE = 00000008  
XWBSW\_STS = 0000000E  
XWBSW\_TIMER = 00000050  
XWBSW\_TIM\_ID = 00000048  
XWBSW\_TIM\_INACT = 0000004C  
XWBSW\_X\_REASON = 00000046  
XWBSZ\_NDC = 00000084  
ZNABUF = 0000011C R  
\_SS\_ = 00000000 02

+-----+  
! Psect synopsis !  
+-----+

PSECT name	Allocation	PSECT No.	Attributes
ABS	00000000 ( 0.)	00 ( 0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	00000000 ( 0.)	01 ( 1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
NET_IMPURE	00000130 ( 304.)	02 ( 2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
NET_PURE	00000082 ( 130.)	03 ( 3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC LONG
NET_CODE	0000096A ( 2410.)	04 ( 4.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC LONG
NET_LOCK_CODE	0000012C ( 300.)	05 ( 5.)	NOPIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC BYTE

+-----+  
! Performance indicators !  
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	34	00:00:00.05	00:00:00.31
Command processing	129	00:00:00.98	00:00:02.95
Pass 1	741	00:00:29.71	00:00:49.52
Symbol table sort	0	00:00:04.21	00:00:07.91
Pass 2	631	00:00:06.49	00:00:14.79
Symbol table output	56	00:00:00.40	00:00:00.96
Psect synopsis output	6	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1599	00:00:41.88	00:01:16.49

The working set limit was 900 pages.

163425 bytes (320 pages) of virtual memory were used to buffer the intermediate code.

There were 160 pages of symbol table space allocated to hold 2818 non-local and 95 local symbols.

1563 source lines were read in Pass 1, producing 32 object records in Pass 2.

65 pages of virtual memory were used to define 53 macros.

+-----+  
! Macro library statistics !  
+-----+

Macro library name	Macros defined
-\$255\$DUA28:[SHRLIB]NMALIBRY.MLB;1	1
-\$255\$DUA28:[SHRLIB]EVCDEF.MLB;1	0
-\$255\$DUA28:[NETACP.OBJ]NETDRV.MLB;1	1
-\$255\$DUA28:[NETACP.OBJ]NET.MLB;1	17
-\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	6
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2	19
TOTALS (all libraries)	44

3081 GETS were required to define 44 macros.

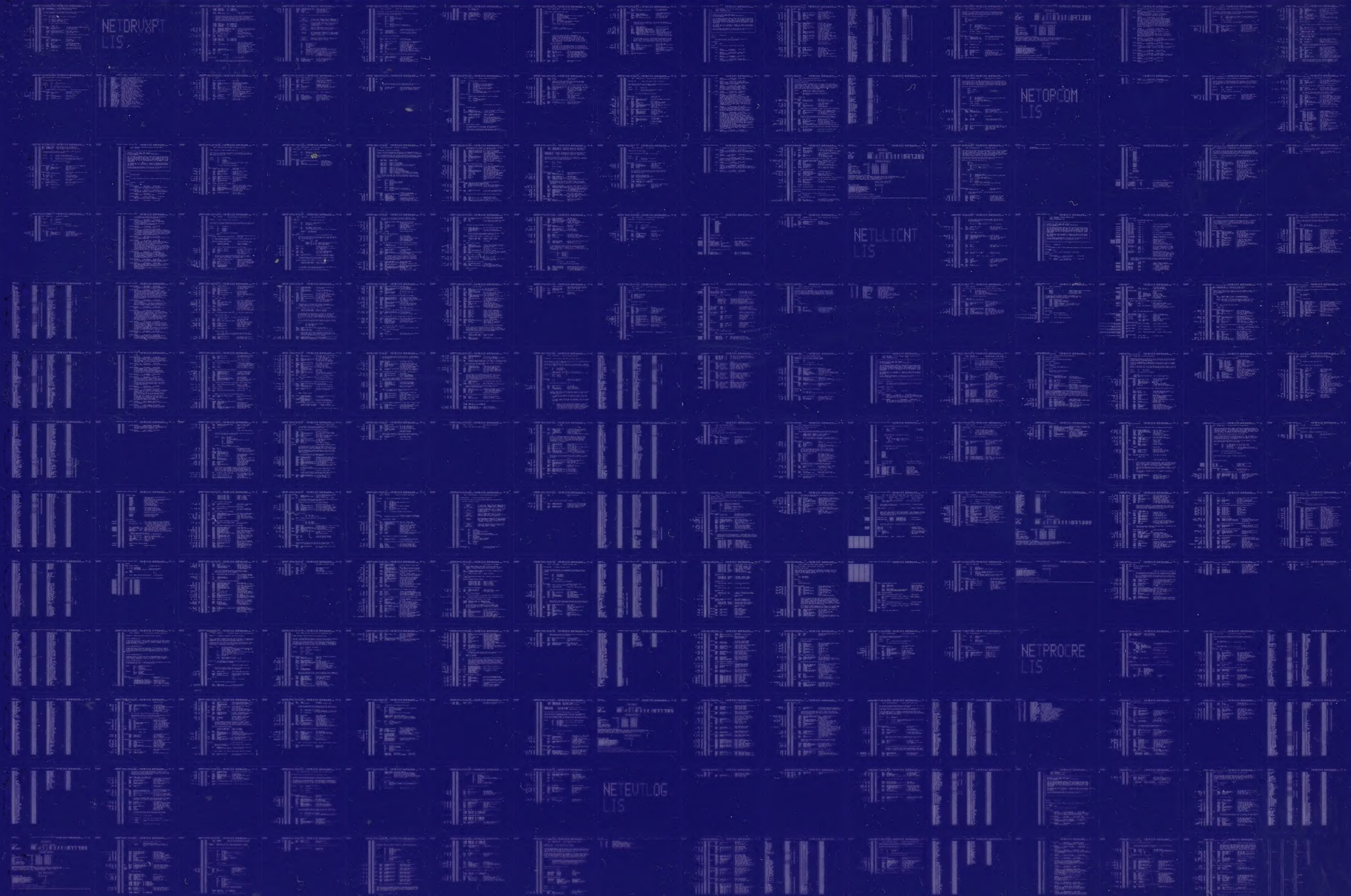
There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:NETPROC/OBJ=OBJ\$:NETPROC MSRC\$:NETPROC/UPDATE=(ENH\$:NETPROC)+EXECML\$/LIB+LIB\$:NET/LIB+LIB\$:NETDRV/LIB+SHRLIB\$



0278 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY





0279 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY